Securing IEEE 802.11 Wireless LANs

by

David Ross

Bachelor of Engineering (Electrical) *(University of Queensland)* – 1986
Bachelor of Arts (Computer Science) *(University of Queensland)* – 1996
Postgraduate Bachelor of Arts Honours (CS) *(University of Queensland)* – 1999

Thesis submitted in accordance with the regulations for
the Degree of Doctor of Philosophy

Information Security Institute
Faculty of Science and Technology
Queensland University of Technology

June 7, 2010

Keywords

802.11, IEEE 802.11, 802.11i, 802.11w, wireless, wireless LAN, WLAN, wireless networks, network security, information security, data security, privacy, Wired Equivalent Privacy, WEP, Wireless Ethernet Compatibility Alliance, WECA, Wireless Fidelity Alliance, Wi-Fi, Wi-Fi Protected Access, WPA, WPA2, Temporal Key Integrity Protocol, TKIP, CCMP, transition security network, TSN, robust security network, RSN, robust security network association, RSNA, data link, link layer, MPDU, PSDU, address encryption, pervasive, ubiquitous.

Abstract

As the acceptance and popularity of wireless networking technologies has proliferated, the security of the IEEE 802.11 wireless local area network (WLAN) has advanced in leaps and bounds. From tenuous beginnings, where the only safe way to deploy a WLAN was to assume it was hostile and employ higher-layer information security controls, to the current state of the art, all manner of improvements have been conceived and many implemented.

This work investigates some of the remaining issues surrounding IEEE 802.11 WLAN operation. While the inherent issues in WLAN deployments and the problems of the original Wired Equivalent Privacy (WEP) provisions are well known and widely documented, there still exist a number of unresolved security issues. These include the security of management and control frames and the data link layer protocols themselves. This research introduces a novel proposal to enhance security at the link layer of IEEE 802.11 WLANs and then conducts detailed theoretical and empirical investigation and analysis of the effects of such proposals.

This thesis first defines the state of the art in WLAN technology and deployment, including an overview of the current and emerging standards, the various threats, numerous vulnerabilities and current exploits. The IEEE 802.11i MAC security enhancements are discussed in detail, along with the likely outcomes of the IEEE 802.11 Task Group W[1], looking into protected management frames. The problems of the remaining unprotected management frames, the unprotected control frames and the unprotected link layer headers are reviewed and a solution is hypothesised, to encrypt the entire MAC Protocol Data Unit (MPDU), including the MAC headers, not just the MAC Service Data Unit (MSDU) commonly performed by existing protocols.

[1]Readers please be aware, irrespective of any final acceptance or publication date on this thesis, this dissertation was written prior to the publication of any output from IEEE 802.11 Task Group W and thus may not necessarily reflect the current information in that respect.

The proposal is not just to encrypt a copy of the headers while still using cleartext addresses to deliver the frame, as used by some existing protocols to support the integrity and authenticity of the headers, but to pass the entire MPDU only as ciphertext to also support the confidentiality of the frame header information. This necessitates the decryption of every received frame using every available key before a station can determine if it is the intended recipient. As such, this raises serious concerns as to the viability of any such proposal due to the likely impact on throughput and scalability. The bulk of the research investigates the impacts of such proposals on the current WLAN protocols. Some possible variations to the proposal are also provided to enhance both utility and speed.

The viability this proposal with respect to the effect on network throughput is then tested using a well known and respected network simulation tool, along with a number of analysis tools developed specifically for the data generated here. The simulator's operation is first validated against recognised test outputs, before a comprehensive set of control data is established, and then the proposal is tested and and compared against the controls. This detailed analysis of the various simulations should be of benefit to other researchers who need to validate simulation results. The analysis of these tests indicate areas of immediate improvement and so the protocols are adjusted and a further series of experiments conducted. These final results are again analysed in detail and final appraisals provided.

Contents

List of Figures

List of Tables

Declaration

Previously Published Material

The following papers have been published or presented, and contain material based on the content of this thesis. The papers are listed in the order of peer-reviewed [1, 2] publications followed by industry presentations [3–5].

[1] David Ross. The Security of Wireless Computing Technologies. In Andrew Clark, Kathryn Kerr, and George Mohay, editors, *AusCERT Asia Pacific Information Technology Security Conference Refereed R&D Stream*, pages 51–63. Australian Computer Emergency Response Team, May 2005. ISBN: 1-86499-799-0.

[2] David Ross, Andrew Clark, and Mark Looi. Securely Deploying IEEE 802.11 WLANs. In A. Clark, M. McPherson, and G. Mohay, editors, *Proceedings of AusCERT Asia Pacific Information Technology Security Conference (AusCERT2007): Refereed R&D Stream*, pages 50–70. Australian Computer Emergency Response Team, University of Queensland, May 2007. ISBN: 978-1-86499-877-1.

[3] David Ross. A Review of Actual IEEE 802.11 Deployment Practices. AARNet Ozeconference, 14th May 2008. Australian Academic and Research Network.

[4] David Ross. Top Reasons for Successful WLAN Penetrations in the Past Year. In *Proc. 16th Australian System Administrators Conference (SAGE-AU'2008)*, Adelaide, Australia, August 2008. The System Administrators Guild of Australia (SAGE-AU).

[5] Luke Turner and David Ross. 802.1X — from business drivers to implementation. Bridge Point Communications 'TechFast' (vendor technical breakfast briefing), February 2009.

Appendix A was posted to the "ns-users" mail list in response to other *ns-2* researchers' requests for assistance with *ns-2* installation on Linux platforms.

Acknowledgements

This work would not have been possible without the support and encouragement of my supervisors Professor Mark Looi and Associate Professor Andrew Clark, whose expert guidance and diligence proved critical in my seeing this undertaking to its conclusion.

I would particularly like to thank my wife, Patricia (Trish) Johansson, who walked into this project shortly after its inception, with no knowledge of either the subject matter or the family sacrifices to come, but nevertheless continually supported, guided, prompted and eventually openly coerced my efforts to completion. Along with Trish, my children, Adrian, Cameron, Nikiesha, Joel and Arabella, were all wonderful in their assistance, support and patience.

I am extremely grateful for the support of Professor Ed Dawson, Director of the Information Security Institute at the Queensland University of Technology through most of this candidature, including his tireless efforts to explain to me the differences between conferences of academic merit and the industry vendor-driven extravaganzas that I was accustomed to.

Some of the research in this thesis was performed in association with other members of the Information Security Institute. I would like to especially acknowledge Rupinder Gill and Jason Smith, for their input over the years, in this and related fields, not the least for their enlightenment on the vagaries of the life of a postgraduate research student. Chapter 3 includes creative endeavours from various pieces collaborative work that I undertook with Professor Mark Looi, Associate Professor Andrew Clark, Rupinder Gill and Jason Smith. Chapter 5 was sparked from a discussion I had one lunchtime with Juan Manuel (Juanma) González Nieto and an obscurely obvious and brilliant out-of-the-box observation by Jason Smith, which triggered a chain of thought over the ensuing weeks that pulled years of meandering research into a single focussed direction.

I would like to thank Professor Eugene Spafford for his time and hospitality,

along with all of his team in CERIAS at Purdue University, especially Marlene Walls and Elisa Bertino, who were so very helpful in arranging the diverse array of research faculty to talk to during my stay there.

In addition, I would like to acknowledge the contribution of many others who have helped this journey by their input, either academic or technical, directly or in conversation. They include:

Ejaz Ahmed (ISI)	Simon Hackett (Internode)	Jason Reid (ISI)
Stig Andersson (ISI)	Anne Hamburger (FIT)	Neville Richter (ISI)
Jason Andrade (AARnet)	Elizabeth Hansford (ISI)	Jared Ring (ISI)
Paul Armstrong (Google)	Yvonne Hitchcock (ISI)	Jaime Romaguera (FIT)
Richard Au (FIT)	John Holford (ISI)	Philip Saunders (HP)
Ken Blinco (BPC)	Jaga Indulska (UQ)	Bradley Schatz (ISI)
Colin Boyd (ISI)	MehdiKianiHarchegani(ISI)	Jodie Siganto (BPC)
Matt Bradford (ISI)	Peter Kootsookos (UQ)	Jason Smith (ISI)
Mark Branagan (ISI)	Barry Lynam (FIT)	Tim Smith (BPC)
Linda Burnett (FIT)	David Lynas (DLCG)	Suriadi (ISI)
Mark Carey-Smith (ISI)	Andrew Marrington (ISI)	Zeeshan Syed (ISI)
Kevin Chen (ISI)	Adrian McCullagh (ISI)	Suratose Tritilanunt (ISI)
Raymond Choo (ISI)	Lois McLaughlin (QUT)	Ricky Tunny (FIT)
David Conran (Google)	Veena Merz (FIT)	John Vallickad (FIT)
Elspeth Dalton (QUT)	George Mohay (ISI)	Jason Weiss (FIT)
Robert Dawson (ISI)	Lee Monette (SAGE-AU)	Jan Wilcox (FIT)
Lesley Dibley (ISI)	Christine Orme (ISI)	On Wong (FIT)
Katie Duczmal (UQ)	Sorot Panichprecha (ISI)	Charles Woo (ISI)
Ernest Foo (ISI)	Marius Portmann (UQ)	Jimmy Xu (ISI)
Gary Gaskell (ISI)	Janne Rayner (QUT)	Jacob Zimmermann (ISI)
Praveen Gauravaram (ISI)	Glen Reeve (UQ)	Elizabeth Zwicky
Juan González Nieto (ISI)	David Reeves (UQ)	(Great Circle Associates)

I sincerely apologise to anyone I may have forgotten to include.

Acronyms and Abbreviations

The following acronyms and abbreviations are used in this document in the context as defined herein.

3G — 3rd Generation (wireless services) (developed by the 3GPP)
3GPP — 3rd Generation Partnership Project (for wireless telephony)
AAA — Authentication, Authorisation and Accounting
AAD — Additional Authentication Data
ACA — Australian Communications Authority (now ACMA)
ACK — ACKnowledgment
ACL — Access Control List
ACMA — Australian Communications and Media Authority (formerly ACA)
AES — (NIST) Advanced Encryption Standard (FIPS 197)
AES-CCMP — AES in CCM Protocol
AID — Association IDentifier
AP — (wireless network) Access Point
ARP — Address Resolution Protocol
AS — Authentication Server
ASN.1 — Abstract Syntax Notation One (ITU-T X.208)
AusCERT — Australian Computer Emergency Response Team
BER — Basic Encoding Rules (ITU-T X.209) (of the ASN.1)
BSA — Basic Service Area
BSS — Basic Service Set
BSSID — Basic Service Set IDentification
CBC — Cipher-Block Chaining
CBC-MAC — Cipher-Block Chaining with Message Authentication Code
CBR — Constant Bit Rate (traffic source)
CCA — Clear Channel Assessment

CCITT — Comité Consultatif International Téléphonique et Télégraphique (International Telegraph and Telephone Consultative Committee, now ITU-T)
CCM — CTR with CBC-MAC
CCMP — (AES in) CTR with CBC-MAC Protocol
CDMA — Code Division Multiple Access
CERT — Computer Emergency Response Team
CERT/CC — CERT Coordination Center
CF — Contention Free
CFP — Contention-Free Period
CHAP — Challenge Handshake Authentication Protocol
COTS — Commercial Off-The-Shelf (items/software/equipment)
CP — Contention Period
CRC — Cyclic Redundancy Check
CS — Carrier Sense
CTR — CounTeR mode (of use of a cipher)
CTS — Clear To Send
CW — Contention Window
DCF — Distributed Coordination Function
DECT — Digital Enhanced Cordless Telecommunications (formerly Digital European Cordless Telecommunications)
DER — Distinguished Encoding Rules (a unique subset of the BER)
DHCP — Dynamic Host Configuration Protocol
DIFS — Distributed (coordination function) InterFrame Space
DMZ — De-Militarised Zone
DNS — Domain Name System
DoS — Denial of Service
DS — Distribution System (to interconnect a set of BSSs to create an ESS)
DSS — Distribution System Service
DSD — Defence Signals Directorate (Australian Department of Defence)
DSSS — Direct Sequence Spread Spectrum
EAP — Extensible Authentication Protocol (IETF RFC 3748)
EAPOL — Extensible Authentication Protocol over LANs (IEEE Std 802.1X)
EAPOW — Colloquial term for EAP over WLAN (non-standard)
EIFS — Extended InterFrame Space
EMR — ElectroMagnetic Radiation

ESS — Extended Service Set

FHSS — Frequency-Hopping Spread Spectrum

FIPS — (USA) Federal Information Processing Standard

GMK — Group Master Key

GPRS — General Packet Radio Service

GSM — Global System for Mobile communications
(formerly Groupe Spécial Mobile)

GTK — Group Temporal Key

GTKSA — Group Temporal Key Security Association

GUI — Graphical User Interface

HA — High Availability (service/configuration/requirement)

IBSS — Independent Basic Service Set

ICMP — Internet Control Message Protocol

ICV — Integrity Check Value

IDS — Intrusion Detection System

IEEE — Institute of Electrical and Electronics Engineers, Inc.

IETF — Internet Engineering Task Force

IFS — InterFrame Space

IP — Internet Protocol

IPS — Intrusion Prevention System

IrDA — Infrared Data Association

IR — InfraRed

IS — Interim Standard (EIA/TIA)

ISI — Information Security Institute[2] (at QUT, formerly ISRC)

ISIG — Information Security Interest Group (Australia)

ISM — Industrial, Scientific, and medical

ISO — International Organisation for Standardisation (not an acronym[3])

ISRC — Information Security Research Centre (at QUT)

IT — Information Technology

IT&T - Information Technology and Telecommunications

ITU — International Telecommunication Union

[2]Note that ISI is also used by the Information Sciences Institute, the current home of the *ns-2* simulator at the University of Southern California, however, although *ns-2* is used extensively in this work, this dissertation only uses the term ISI to refer to the authors affiliation, the Information Security Institute at QUT.

[3]ISO is derived from the Greek isos, meaning ‘equal’ — acronyms would differ according to language (‘IOS’ in English, ‘OIN’ in French for Organisation Internationale de Normalisation).

ITU-T — ITU Telecommunication Standardization Sector
IV — Initialisation Vector
kb — kilobit (1000 bits) c.f. Kib
kB — kilobyte (1000 bytes) c.f. KiB
KCK — EAPOL-Key Confirmation Key
KEK — EAPOL-Key Encryption Key
Kib — Kibibit (1024 bits) c.f. kb
KiB — Kibibyte (1024 bytes) c.f. kB
LAN — Local Area Network
Layer 2 — Refers to the Data Link Layer of the ISO OSI model
Layer 3 — Refers to the Network Layer of the ISO OSI model
Layer 4 — Refers to the Transport Layer of the ISO OSI model
LIPD — (Australian) Low Interference Potential Devices (Class Licence)
MAC — Medium Access Control ***or*** Message Authentication Code
MAN — Metropolitan Area Network
MBOA — MultiBand OFDM Alliance (for IEEE Std 802.15.3a)
Mgt — Management (service/server/function)
MIB — Management Information Base
MIC — Message Integrity Code
MIMO — Multiple Input Multiple Output
MMS — Multimedia Message Service
MS — Mobile Station — mobile telephone handset
msb — most significant bit
MS-CHAP — MicroSoft Challenge Handshake Authentication Protocol
N/A — Not Applicable
NAC — Network Access Control
NIC — Network Interface Card
NII — (USA-specific) National Information Infrastructure (radio frequency band), also U-NII
NIST — (USA) National Institute of Standards and Technology
NTP — Network Time Protocol (IETF RFC 1305)
OFDM — Orthogonal Frequency Division Multiplexing
OSI — (ISO) Open Systems Interconnection (Basic Reference Model — "OSI Reference Model")
OTP — One-Time Password (single-use)

PAN — Personal Area Network

PAP — Password Authentication Protocol

PC — Point Coordinator

PCF — Point Coordination Function

PDA — Personal Digital Assistant

PDU — Protocol Data Unit

PHY — PHYsical (layer)

PIFS — Point (coordination function) InterFrame Space

PKI — Public Key Infrastructure

PMK — Pairwise Master Key

PMKID — Pairwise Master Key IDentifier

PMKSA — Pairwise Master Key Security Association

ppm — parts per million

PPM — Pulse Position Modulation

PRF — Pseudo-Random Function

PRNG — Pseudo-Random Number Generator

PSK — PreShared Key

PTK — Pairwise Transient Key

PTKSA — Pairwise Transient Key Security Association

QoS — Quality of Service

QUT — Queensland University of Technology (Australia)

RADIUS — Remote Authentication Dial-In User Service (IETF RFC 2865)

RF — Radio Frequency (EMR)

RFC — Request For Comments

RSN — Robust Security Network

RSNA — Robust Security Network Association

RSSI — Received Signal Strength Indication

RTS — Request to Send

RX — Receive ***or*** Receiver

SAGE-AU — Systems Administrators Guild of Australia

SIFS — Short InterFrame Space

SIM — Subscriber Identity Module

SMS — Short Message Service ***or*** (Microsoft) Systems Management Server

SMTP — Simple Mail Transfer Protocol

SNMP — Simple Network Management Protocol

ssh — Secure Shell (program and sshd daemon)
SSID — (IEEE 802.11) Service Set IDentifier
SSL — Secure Sockets Layer
STA — STAtion — any device that contains an IEEE 802.11 conformant MAC and PHY interface to the WM
STP — Shielded Twisted Pair (cable)
SYN — SYNchronise (TCP)
TACACS — Terminal Access Controller Access Control System
TCP — Transmission Control Protocol
TCP/IP — Transmission Control Protocol over Internet Protocol
TDMA — Time Division Multiple Access
TIA — Telecommunication Industries Association
TKIP — Temporal Key Integrity Protocol
TSN — Transition Security Network
TX — Transmit ***or*** Transmitter
UA — (VeriSign) Unified Authentication (product)
U-NII — (USA-specific) unlicensed National Information Infrastructure (radio frequency band)
UQ — University of Queensland (Australia)
URL — Universal Resource Locator
USA — United States of America
UTP — Unshielded Twisted Pair (cable)
UWB — Ultra Wideband
VLAN — Virtual Local Area Network
VoIP — Voice over Internet Protocol
VPN — Virtual Private Network
VTP — VLAN Trunking Protocol
WAN — Wide Area Network
WAP — Wireless Application Protocol
WDS — Wireless Distribution System
WEP — Wired Equivalent Privacy
WEP-40 — WEP using a 40-bit key (64-bit[4] WEP)
WEP-104 — WEP using a 104-bit key (128-bit[5] WEP)
WIC — (Cisco) WAN Interface Card

[4]Some vendors refer to WEP-40 as 64-bit WEP (including 24 bits of IV).
[5]Some vendors refer to WEP-104 as 128-bit WEP (including 24 bits of IV).

Wi-Fi — (IEEE 802.11 series technologies, certified by the) Wireless Fidelity (Alliance)
WLAN — Wireless LAN
WM — Wireless Medium
WPA — Wi-Fi Protected Access (certification by the Wi-Fi Alliance, for a TSN)
WPA2 — Wi-Fi Protected Access 2 (certification by the Wi-Fi Alliance, for a RSN)
WPAN — Wireless PAN

Chapter 1

Introduction

As society unreservedly accepts ubiquitous computing, vendors are releasing bleeding-edge pervasive computing technologies to a hungry consumer market which has little appreciation, or concern, for the inherent risks and associated consequences that these technologies may introduce. Many of these new technologies involve electromagnetic broadcast communication protocols used in uncontrolled and potentially hostile environments.

This impetus to take up new technology often leads to immature designs or hastily conceived deployments exposing inadvertent vulnerabilities that lead to security failures.

Consumers are embracing such wireless communication technologies at an unprecedented rate. After James Clerk Maxwell postulated the existence electromagnetic waves to the Royal Society of London in 1864[1] [1], some 23 years elapsed before Heinrich Hertz demonstrated the production of such electromagnetic waves in 1887 [2]. It took nearly a further decade before either Nikola Tesla or Guglielmo Marconi publicly demonstrated the practical transmission of electromagnetic signals [3, 4]. Today, new wireless communication technologies are presented to consumers virtually on a monthly basis — albeit often just different flavours of the same precarious concept — as vendors attempt to cajole an unsuspecting market to take up their particular offering with sufficient zeal to render it a de-facto standard, or at least to pay for the research and development costs before a de-jure standard obsoletes their product.

[1]Published in 1865 [1].

This work investigates some of the remaining issues in the operation of *IEEE Std 802.11* [5] wireless LANs (WLANs). These include the security of the management frames and control frames and the data link layer protocols themselves.

This research introduces a novel proposal to enhance security of IEEE 802.11 WLANs by using cryptographic protection of the link layer. The proposal introduces significant implications for the general utility and throughput of the protected WLANs and this work then investigates the viability of this proposal with respect to the impacts on throughput and scalability.

Section 1.1 describes the research problem and section 1.2 details the goals of this research and how this research relates to published work in the same field. Following this, section 1.3 presents the main outcomes of this research and then section 1.4 outlines the rest of this dissertation. Finally, section 1.5, defining the conventions used throughout the remainder of this document, concludes this introduction.

1.1 Research Problem

Due to their broadcast mode of transmission and unbounded electromagnetic transmission medium, all wireless communications technologies are inherently insecure in every aspect of information assurance: confidentiality, integrity and availability. Combining this with vendor drivers of time-to-market, cost-of-development and cost-of-production; along with commercial competition, business expediency and consumer naivety; and these inherent insecurities are often realised. In the rush to bring products to market and standards to industry, the primary issues have not yet been satisfactorily resolved.

There is currently a dramatic increase in the utilisation of wireless bandwidth in the various unlicensed spectra [6, 7] and this rapid public adoption [8] of unseasoned, often provisional, wireless technologies results in an array of information security issues, described in detail in the current literature considered in Chapter 2.

In the case of IEEE 802.11 [5] wireless LANs, the Wired Equivalent Privacy (WEP) [9, pp. 61–65] protocol was intended to provide security properties similar to that of wired networks and is still the most commonly deployed built-in security mechanism in Australia [10, 11]. WEP is easily cracked (defeated) using commonly available software detailed in Chapter 2.

The *IEEE Std 802.11i* [12] amendment [Chapter 3, subsection 3.2] was developed to address the critical security issues of preceding IEEE 802.11 WLANs. However, an IEEE 802.11i Robust Security Network (RSN), using the Advanced Encryption Standard (AES) in CCM[2] Protocol (CCMP), requires new hardware to operate and so deployments in legacy environments will typically fall back to using the Temporal Key Integrity Protocol (TKIP), where "the confidentiality and integrity mechanisms are not as robust as those of CCMP" [13, p. 2], or even to a Transition Security Network (TSN) that allows pre-RSN associations, as described in Chapter 3.

Moreover, the *IEEE Std 802.11i* only improves the MAC security for data frames and provides no protection at all for WLAN control frames or management frames or the data link layer protocols themselves. This lack of protection for management and control frames still provides multiple attack vectors into these networks.

Today's state-of-the-art WLANs remain vulnerable to trivial traffic analysis, using information gained from the unprotected management frames and link layer headers; selective Denial of Service (DoS), through the manipulation of either unprotected control frames or unprotected management frames, using information gained from the unprotected data link headers; or even certain Man-In-The-Middle (MITM) attacks, involving selective disassociation, through the manipulation of unprotected management frames, and a spoofed Access Point (AP) offering reduced security, using more unprotected management frames and other information gained from legitimate unprotected data link headers.

The details of the coming amendment from IEEE P802.11w[3] for Protected Management Frames, as discussed in Chapter 4 [subsection 4.7], are as yet unconfirmed, but are likely to provide only protection for a subset of WLAN management frames — and then only after the IEEE 802.11i key negotiations are complete — still leaving a number of management frames, all control frames and all of the data link layer protocol information unprotected.

[2]CTR (CounTeR mode) with CBC-MAC (cipher-block chaining (CBC) with message authentication code (MAC)) [12, p. 5].

[3]Readers please be aware, irrespective of any final acceptance or publication date on this thesis, this dissertation was written prior to the publication of any output from IEEE 802.11 Task Group W and thus may not necessarily reflect the current information in that respect.

1.2 Research Goals

The goal of this research is to investigate possible enhancements to the existing and proposed protocols to provide additional protections in these areas while minimizing additional loss of general utility for IEEE 802.11 WLANs as a whole.

This research proposes a novel solution to protect all management frames, all control frames and all of the components at the data link layer by encrypting the entire MAC Protocol Data Unit (MPDU), including the MAC headers, but not the Frame Correction Sequence (FCS), to provide both integrity and confidentiality. Unlike existing schemes, this proposal is not only to protect the MAC Service Data Unit (MSDU), or to protect a copy of the MPDU header detail while still using cleartext addresses, but to pass only ciphertext for the MPDU, with a cleartext FCS.

Due to the difficulties of processing encrypted addressing information, where many different possible pairwise keys must be tried for each frame received in order to determine if it is intended for that particular station, the proposed solution foreshadows a considerable risk of severely adversely affecting network throughput. While numerous issues, such as increased complexity, key management and implementation considerations, may render this proposal unsuitable for particular applications, a significant impact to network throughput is the greatest threat to the overall viability of the general solution. A number of variants of the proposal are suggested in mitigation for this and various other possible issues. This work examines the extent of likely throughput impacts and investigates the viability of implementing such a solution across an extensive range of WLAN configurations.

1.2.1 Research Objectives

The required research objectives determined to achieve these goals included:

- Review of the current literature on wireless network security to provide a background in the work;
- Establish the current state of the art for WLAN security with respect to the emerging implementations of TSNs and RSNs under the *IEEE Std 802.11i* ratified during the course of this research;

- Analyse any implications for current WLAN security in view of the current amendments, along with the possible implications of the emerging standards, including the forthcoming amendment from IEEE P802.11w[3] for Protected Management Frames, to determine the remaining level of protection afforded to WLANs and to confirm the major areas of remaining vulnerabilities;

- Present the proposed avenues to resolve some of these remaining security issues and discuss their implications for WLAN utility and performance;

- Develop methods to test these proposals in light of their possible impacts on the general utility of WLANs, including methods to measure acceptable metrics to provide usable empirical data for the research community by which to assess the viability of such proposals;

- Conduct credible simulations of implementations of each of these proposals with the necessary rigour to provide competent evidence on the likely impacts of applying such amendments to existing architectures, revising any tools as required; and

- Finally to analyse the results so obtained, and provide an assessment of the probable impacts of the actual deployment of such protocols in usable networks and to indicate future directions to develop this knowledge base.

1.2.2 Relationship to Published Research in the Field

There is considerable published work on the insecurities of wireless networking in general and extensive publication on the specific security issues of WLANs using WEP, detailed in the current literature in Chapter 2.

There are also the various works covering issues with the IEEE 802.11i amendment and the remaining insecurities in current networks, as detailed in Chapter 3, providing the state of the art for IEEE 802.11 WLAN security and the current deployments of Wi-Fi Protected Access (WPA) and WPA2. This includes extensive publication on the failures in the security of control and management frames and the resulting impacts that these cause, the issues of TSNs, direct attacks on the Pre-Shared Key (PSK) modes of operation and the issues of weak configurations permitted by many vendor offerings with mixed modes of operation.

While many protocols encrypt MPDU payloads, such as the entire MSDU, or higher-layer PDUs or their respective payloads [14–21], and often also include a message integrity check (MIC) protecting the MAC headers (and any encryption headers) themselves [15], as well as frame source authenticity [19, 20], the author has been unable to locate any examples of cryptographic confidentiality protection of link layer communications that include the MAC layer headers.

It is noted that many protocols also include copies of the MAC addresses and other MAC header data within the encrypted payload, such as MACsec [20] (also referred to by the project name LinkSec), but cleartext data link layer headers are still transmitted in these cases. As such, these schemes may be able to protect the integrity of the headers, but cannot protect their confidentiality. While these headers may not appear to contain a significant amount of information, their exposure greatly aids any forgery attempts and also assists in traffic analysis and, where related, cryptanalysis of the ciphertext payload. Tunnelling protocols only provide confidentiality of the tunnelled addresses, leaving the link layer data of the tunnel itself exposed.

There also exist schemes involving encrypted MAC addresses for the purposes of cataloguing or registration, such as in Yang and Liu [22], but these are not designed for, and do not provide, communications protection.

1.3 Outcomes of the Research

The principal outcome of this research is to show that the overheads introduced by encrypting the MAC addresses in WLAN frames will not adversely affect the network throughput under reasonable conditions — even where an AP may have to attempt decryption of each frame with hundreds of different pairwise keys in order to determine the correct key in use. This research demonstrates that cryptographic protection of the entire MPDU is viable for WLAN and thus provides essential support for the development of schemes, such as the proposal and variants presented in this work, to enhance WLAN security.

Such schemes provide increased protection, beyond the current state of WLAN security, from the types of targeted DoS and MITM attacks described in Chapters 2, 3 and 4; and even attacks using the legitimate MAC information, unprotected control frames, or the remaining unprotected management frames from the emerging work on Protected Management Frames, as discussed in Chapter 4.

Specific outcomes of this research include:

1. A review of the current literature on IEEE 802.11 wireless networking, presented at the 2005 *AusCERT Asia Pacific Information Technology Security Conference* [23] and provided in Chapter 2;

2. A summary of the current state of the art of IEEE 802.11 WLAN security including the *IEEE Std 802.11i* MAC Security Enhancements, their implementation and the dependencies and differences between a TSN, RSN, WPA and WPA2, presented in part at the 2007 *AusCERT Asia Pacific Information Technology Security Conference* [24] and provided in Chapter 3;

3. An analysis of some of the remaining security issues, unresolved by the existing or proposed protocols, provided in Chapter 4;

4. A proposal to enhance the security of wireless networks at the link layer, by cryptographically protecting the MPDU for all traffic, including the entire MAC header, for confidentiality as well as integrity, enhancing data link security beyond any existing or proposed protocols, provided in Chapter 5;

5. A set of tools to analyse *cmu-trace* wireless trace file data from the well-known *ns-2* simulator, provided in Chapter 6, including:

   - General-purpose *awk* analysis scripts capable of handling the default *ns-2.33 cmu-trace* wireless trace file format (as opposed to readily available scripts for the more verbose "new wireless trace" file format);
   - Special-purpose (project-specific) analysis tools to interpret the simulations configured here and to discern multi-station traffic interaction, determined from the default *ns-2.33 cmu-trace* wireless trace files without requiring any modification to either the simulator itself or the simulation configurations, including:
     – Individual station choices of number of slots for backoff periods,
     – Individual frame transmission times,
     – Individual frame propagation times,
     – Inter-frame delays,
     – Inter-layer delays, and
     – Intra-layer delays;

6. Detailed simulation data demonstrating the effects and impacts of applying cryptographic techniques to low-level protocols in the IEEE 802.11 stack, available[4] as a compressed tar archive (tarball), including: comprehensive explanations of the necessary modifications to the standard CMU/Rice `Mac/802_11` [25] *C++* class files within the *ns-2* simulator, as provided in Chapters 6, 7 and 8; complete configurations for all tests performed, as detailed in Chapters 7 and 8; the capture of all the data produced by every test;

7. Results of the effects of these proposals and extensive analysis of each of the simulations, providing insight, not only into the protocols proposed here, but also in general for any proposals presenting timing-related impacts to standard IEEE 802.11 MAC protocols, provided in Chapters 7 and 8; and

8. A summary of the effects and viability of a low-level protocol protection in IEEE 802.11 networks, along with avenues of further work to advance this knowledge, provided in Chapter 9.

1.4 Thesis Structure

The remainder of this chapter details the thesis structure and the conventions used throughout this document.

Following this, Chapter 2 provides the background information from the current literature on wireless networking in general, IEEE 802.11 WLANs, the various standards, the initial IEEE 802.11 security provisions, the issues with those provisions and the initial attempts to improve WLAN security.

Chapter 3 describes the current state of the art for IEEE 802.11 wireless network security, including the recent amendments for the MAC security enhancements and how they are implemented in Wi-Fi Protected Access (WPA) and WPA2. The differences between a TSN and an RSN and and some of the pitfalls of the implementations are discussed, including analysis of various weak configurations of WPA and WPA2, where attempts to implement an RSN fail or lead to a TSN, as well as direct attacks on pre-shared key modes and TKIP.

[4]Detailed simulation data and the *ns-2* code modifications are available from the author, David Ross <dave@antacs.com>, or supervisors, Professor Mark Looi <m.looi@qut.edu.au> or Associate Professor Andrew Clark <a.clark@qut.edu.au>.

Chapter 4 provides more detailed operational information and examines the current and emerging IEEE 802.11 network technologies and the potential for malicious activities. This establishes the need for this research and considers the various applications it has in securing WLANs.

Chapter 5 describes the proposal for Wireless Link Security and discusses the needs of the link layer and the effects of encrypted MAC addresses on both friend and foe. It then introduces alternative algorithms for greater speed, efficiency or operational necessity. The protocol design is investigated, including the operation of the mutable fields, hardware components, the whitening functions and key establishment, including pairwise, group and other broadcast keys and then the deployment needs and the tests required are discussed.

Chapter 6 describes the development of the tools to test the viability of the proposed protocols, including selection of tools and platforms, configurations and validation of the various tools for simulation, analysis and reporting.

Chapter 7 details the actual tests undertaken, including the development of the controls, the effects of multi-station contention, determining the average throughput and establishing control data for UDP and TCP traffic both with and without contention. The control data is then analysed to validate it against expected results both in theory from the *IEEE Std 802.11* and in practice from other research data. In particular, the delays due to normal operation of IEEE 802.11 networks requiring acknowledged directed traffic and the default Distributed Coordination Function (DCF) backoff are analysed and discussed. Then the proposals are tested, first detailing how the tools were modified to implement the modified protocols. The initial results were not as expected and the tools were then modified to ensure that this was not due to inaccuracy within the simulation itself. However, results after the additional modifications merely confirmed the initial results and the tests were continued to forge a comprehensive range of excessive traffic situations under both WLS and PWLS. These results are then analysed and discussed.

From these results, Chapter 8 details modifications to enhance the simulations and draws fresh results from these new simulations, which are then analysed and assessed in detail.

Finally Chapter 9 concludes the work, with a summary of the overall evaluation of the hypothesis, as well as defining a number of areas of future endeavour in this field.

1.5 Conventions Used in This Document

As shown in the acronyms and abbreviations in the front matter, throughout this document the lower-case symbol ‘b’ is used for bits, while the upper-case symbol ‘B’ is used for bytes.

Acronyms ending with or containing a lower-case ‘s’ indicate the plural expansion of the appropriate part of the acronym, e.g. APs for Access Points.

This text is written in Australian English and besides the commonly-known use of ‘-is-’ infixes where others may use ‘-iz-’ and the use of Australian spellings such as the Middle English (from Old French) ‘centre’ or ‘litre’ where others may use ‘center’ or ‘liter’, other localisations which may lead to confusion include ‘full stop’ $\equiv$ ‘period’ and ‘mobile phone’ $\equiv$ ‘cell phone’.

Irrespective of the aforementioned, organisational names herein are spelled[5] verbatim, such as the “International Organization for Standardization”[6] or the “Center for Education and Research in Information Assurance and Security”[7].

Common information technology industry terms, such as the adjective form “conformant” of the recognised noun ‘conformance’ and conjunctions “chipset”, “plaintext”, “ciphertext” and the like, although often not recognised by lexicographers, are used throughout this dissertation without additional definition or hyphenation.

Terms commonly used in IEEE standards or IETF RFCs and unambiguously understood within this field, such as “backoff”, “interframe” and “unicast”, irrespective of whether they appear in any formal lexicon, are used herein without further reference.

In accordance with normal practice [27–29], the full stop has been omitted from the end of any sentence that terminates with any form of path name, including URLs [27].

Symbols for units of measure are written the standard Roman font, always in upright type, in accordance with [30], so as to differentiate these from the italic type used for names, variables and other LaTeX emphasis. This is “applied independent of the font used for surrounding text” [30, pp. 121, 130].

[5] Both “spelled” and “spelt” are listed for the past participle of “spell” in the Second Edition of the Australian Oxford Dictionary [26], although the dictionary itself uses “spelled”.

[6] http://www.iso.org/

[7] http://www.cerias.purdue.edu/

Chapter 2

Wireless Networks

This chapter provides the necessary background information from the current literature on wireless networking, as particularly relevant to IEEE 802.11 WLANs. It describes the basic concepts and maps out the various IEEE 802.11 standards, both issued and proposed, and where they fit into the overall WLAN picture. It includes discussions on the security provisions of the original IEEE 802.11 WLANs and the various issues with those provisions discovered in deployment, along with the initial attempts and methodologies for improving WLAN security.

This wireless networking background derives from the literature review performed by the author at the commencement of this research, significant portions of which were presented by the author in "*The Security of Wireless Computing Technologies*" [23], at the *AusCERT Asia Pacific Information Technology Security Conference (AusCERT2005)*, Gold Coast, Australia, in May 2005. Significant portions of that paper have also been reproduced verbatim in documents by one of the Australian state governments.

This chapter not only sets the scene for the remainder of the dissertation, but establishes the baseline security standard for WLANs that existed at the beginning of this research. The subsequent developments in WLAN security during the course of this research and the current security issues are given in the next chapter, Chapter 3, and the emerging enhancements, at time of writing, are considered with the description of the remaining problems relevant to this research in Chapter 4.

2.1 Wireless Ubiquity

The use of wireless technologies for data communications is proliferating at an unprecedented rate [23]. Cordless telephone handsets utilise various microwave spectra to carry their signals to their wired base stations [31]. Mobile telephone handsets — mobile stations (MS) — maintain regular communications with the cells around them irrespective of whether they are being actively used or not. Increasingly, these handsets are capable of operating over multiple network protocols, designed to swap between the various cellular, wide-area and local-area wireless network protocols, as appropriate, for utility, efficiency or economy [32]. Such devices typically continually scan the air around them searching for the strongest or cheapest connection to use, including, where available, Voice over IP (VoIP) over WLAN (VoFi[1]). Laptop computers and personal digital assistants (PDAs) use the infrared or microwave spectra to communicate with each other or with printers, mobile telephones or wired-infrastructure networks and the Internet.

Wireless networks are deployed in all manner of locations, including [33] homes, schools, university campuses and *Wi-Fi*[2] *hot spots*[3] in cafés, book stores, libraries, shopping centres, public bars and even aircraft [34]. The author has conducted security assessments of wireless deployments in numerous situations, including public transport depots, airport lounges, hotels and commercial offices, as well as a public network in a major art gallery to disseminate information on the works being viewed [11, 35]. The author has also been involved with wireless network security assessments in shared auditoriums where the wireless infrastructure is also shared, by government officers and university students; and even in work for the Australian Federal Parliament House[4] for use not only by the sitting members on the X-IN-CONFIDENCE network but simultaneously by visitors on an UNCLASSIFIED network, both in the same location.

"In some remote communities, WLANs are implemented as a viable *last-mile* technology" [33, p. 44] by users lacking landline DSL or cable services [38].

Wi-Fi devices have been adapted for disaster use, forming an ad hoc network for emergency data transfer, such as with the Wireless Internet Information System for Medical Response in Disasters (WIISARD) [39].

[1]Voice over Wi-Fi
[2]Wireless Fidelity Alliance. URL — http://www.wi-fi.org/
[3]Either free or for a charge
[4]UNCLASSIFIED details released to public tender on 28th November 2007 [36, 37].

While there are a great many existing applications and deployments of wireless networking technology in many varied locations, to expand the utilisation of wireless technologies to critical services or in defence or intelligence environments will require the addition of security features beyond those currently existing. A major hurdle in providing such security is the technology's broadcast nature via an ubiquitous medium, providing network access to adversaries outside the physical security controls of the network owner.

The following sections describe those wireless networking technologies that have the capability to impact upon IEEE 802.11 WLANs, leading up to the description of IEEE 802.11 WLAN itself and the various PHY and MAC protocols available. The original WLAN security provisions are then described, along with the threats and vulnerabilities that were present in WLANs at the commencement of this research. The chapter concludes with details of the mechanisms deployed in the initial attempts to mitigate these risks. This work was presented in [23].

2.2 Wireless Networking Technologies

Wireless Networks utilise the infrared, radio or microwave spectra, under the governance of various protocol suites. Of these, commercial broadcast radio and television clearly had the greatest market penetration towards the end of the last millennium. However, now mobile phone ownership outstrips television ownership by almost 2:1 [40]. Besides the various mobile telephony protocols, which do not generally interfere with WLANs (other than for those MS that may dynamically switch between communications protocols, including the WLAN standards), some of the more common wireless networking technologies of relevance here are the Infrared Data Association (IrDA), IEEE 802.15 Bluetooth and Ultra-wideband (UWB) personal area networks (PANs) and the IEEE 802.11 WLANs, all of which share various common spectra, along with a myriad of unrelated radiating devices.

2.2.1 IrDA

IrDA infrared links, using the near-infrared[5] (120–400 THz) spectrum, establish point-to-point connections at close ranges up to at least 1 metre, such as between two laptop devices or between a PDA and a printer or a mobile telephone, where such devices are IrDA capable. The IrDA inaccurately claims "the secure wireless link" [41] based on infrared being principally a line-of-sight transmission technology, thus limiting covert passive or active intrusion to those devices in the immediate vicinity. Relying on these short-range characteristics, IrDA provides no link level security and anyone within the unobstructed minimum 30° to maximum 60° [42] beam can monitor the transmission, including outside glass doors and windows and reflections from surfaces in rooms and vehicles [43].

In one of his SANS papers in 2004, "Security in Wireless Mobile Communications", Dan Reain related a case attributed to James M. Atkinson of a flight to San Jose where 31 laptops had their IrDA port active and 26 of these allowed unrestricted access [43, p. 2].

2.2.2 Mobile Telephones

While mobile telephone handsets do not generally interfere with WLANs, most handsets today at least include Bluetooth capabilities and a few still provide IrDA capabilities, but more and more frequently these handsets are capable of operating over multiple network protocols. Those that are capable of utilising Bluetooth or WLAN, when configured to swap between wireless network protocols, will regularly scan the air around them searching for the strongest or cheapest connection to use. This means regular probes in WLAN spectra if using active methods and added congestion whenever these spectra are chosen.

This will develop more so, as WLANs are often better than 3G telecommunications for many applications. Paul Henry, AT&T Labs division manager of broadband wireless-systems research has stated that "typical 3G users will get performance up to [only] 56 kbits per second" [44, p. 29], after performance hits for distance, congestion and maintaining simultaneous voice quality.

[5]That part of the infrared spectra that is closer to visible light; cf. mid-infrared and far-infrared.

2.2.3 Bluetooth

Bluetooth is a short-range wire-replacement technology [45, Vol. 1, p. 13] that implements a Wireless Personal Area Network (WPAN). It allows up to eight devices to connect together into a 'piconet'. The most frequent use of Bluetooth technology has been for mobile telephones, not only as wireless headsets and hands-free use in cars, but also to synchronise configurations, data or even email with PDAs or PCs [46, 47]. These uses are expanding to cable-free speakers in home theatre systems and Bluetooth headsets for personal stereos and MP3 players [48], although UWB also made a play for market-share on the former.

Bluetooth uses one of the international unlicensed Industrial, Scientific and Medical (ISM) bands at 2.4 GHz, along with a host of other devices including digital cordless telephones and many WLANs. Bluetooth utilises 79 channels in most countries (23 in France, Japan and Spain) at 1 MHz intervals from 2.402 GHz to 2.480 GHz. The following table (Table 2.1), adapted from Mobile-Info.com [49], outlines the spectrum availability worldwide:

Country	*Frequency Range*	*Channels available*
AU, EU & USA	2400 – 2483.5 MHz	2402 to 2480 MHz (79 channels)
Japan	2471 – 2479 MHz	2473 to 2495 MHz (23 channels)
Spain	2445 – 2475 MHz	2449 to 2471 MHz (23 channels)
France	2446.5 – 2483.5 MHz	2454 to 2476 MHz (23 channels)

Table 2.1: Bluetooth worldwide spectrum availability [49]

It uses frequency hopping spread spectrum (FHSS), changing channels up to 1600 times per second in a pre-arranged pattern. FHSS helps to reduce the effects of narrowband interference and also makes it harder to trivially eavesdrop on data. This is because the frequency hopping pattern is set each time by the devices communicating with one another [50].

The IEEE 802.15.1-2002: *Wireless MAC and PHY Specifications for Wireless Personal Area Networks (WPANs)* [51] was published in June 2002 and is based on the most common Bluetooth version 1.1. IEEE 802.15.2-2003: *Recommended Practice for Coexistence of Wireless Personal Area Networks with Other Wireless Devices Operating in Unlicensed Frequency Bands* [52] is designed to mitigate interference with 802.11 wireless LANs.

2.2.4 Ultra Wideband (UWB)

Ultra Wideband (UWB) techniques can be applied to any radio technology and generally are not relevant or are not supposed to impact on WLAN deployments. However any UWB techniques used for any technology that include the WLAN spectra will increase the “background noise” and may affect IEEE 802.11 technologies.

The IEEE Std 802.15.3 High Rate WPAN does use the 2.4 GHz band (not as UWB) and proposals for the IEEE 802.15.3a, sometimes characterised as “Bluetooth on Steroids” [53], do use UWB techniques, designed to replace cables with short-range very-low-power very-high-bandwidth wireless connections [54]. UWB Bluetooth gives “up to 480 megabits per second at 2 [metres] and 110 megabits per second at 10 [metres], which is the maximum nominal range for both ultrawideband and Bluetooth” [55].

Poor commercial take-up and a dead-locked standards process [56] has led to the IEEE 802.15.3a Project Authorisation Request (PAR) being withdrawn in 2006 [57].

At time of writing, most countries are still legislating UWB regulations. The U.S. limits general UWB to the frequency band 3.1–10.6 GHz, except “vehicular radar in the 22–29 GHz band using directional antennas on terrestrial transportation vehicles” [58, p. 4], with “Attenuation of the emissions below 24 GHz is required above the horizontal plane in order to protect space borne passive sensors operating in the 23.6-24.0 GHz band” [58, p. 4]. To date Australia has only approved UWB for ground penetrating radar [59] and 24 GHz short-range vehicle radar [60, p. 4].

2.2.5 ZigBee

At the oposite end of the data-rate scale, *IEEE Std 802.15.4* “ZigBee” (uses the ZigBee lightweight routing protocol) for telemetry and is designed for very low data rates, so as to give long battery life and low device cost [61]. ZigBee operates in the 2.4 GHz ISM band by default, but uses 868 MHz in Europe and 915 MHz in the U.S. and Australia. IEEE Std 802.15.5, approved and publication expected April 2009, is planned to enable other WPAN meshes to be formed at the MAC layer, without needing ZigBee for routing [62].

2.2.6 WiMAX

In some remote communities, WLANs are deployed as the *last-mile* local loop by users lacking landline DSL or cable services [33,38]. This has been a target market for the IEEE 802.16 [63] WiMAX Forum's broadband Wireless MAN protocol applications [64], which can also use the same spectra (although typically not) as WLANs.

2.2.7 IEEE 802.11 WLAN

While the term ***Wi-Fi*** specifically refers to technology certified by the industry-based *Wireless Fidelity Alliance*[6], its use has been generalised to refer to to any of the IEEE 802.11 series of protocols and their implementations [23]. WLANs allow devices to move about with relative freedom and wirelessly connect to other devices or wired-networks, including the Internet, using a short-range radio transceiver. They also significantly reduce the time and resources needed to set up new networks and allow for dynamic ad hoc networks, easily and/or frequently created, modified and discarded, with little or no fixed infrastructure [23]. They offer data rates 50 to 2000 times faster than dial-up, at peak speeds from 11 mbps (million bits per second) to more than 100 mbps, up to 100 metres from a base station [48].

In general, a WLAN consists of two or more wireless-capable devices communicating with each other. These may be all mobile devices, with no central control, forming an ***ad hoc*** network, or one or more of the devices may include a central control function, an Access Point (AP), forming an ***infrastructure*** mode network. The AP relays data between the wireless nodes in the WLAN "and, in most cases, serves as the only link between the WLAN and the wired networks" [33, p. 45].

Today, the Wi-Fi network interface cards (NICs) most commonly found in PCs and laptops conform to one or more of the IEEE 802.11a, b, g or (draft) n specifications. These standards cover a number of different frequency bands and allow for varying data rates, with some of the newest cards able to operate under multiple standards [48].

[6]Formerly the *Wireless Ethernet Compatibility Alliance* (WECA) [65]

IEEE 802.11 WLAN Topologies

This last subsection of the wireless networking technologies describes WLANs themselves as a grounding for any of the material introduced later in this work. The basic WLAN concepts will be described here, beginning with their logical structures and then those operational aspects that are needed for the WLAN security discussions in later sections.

The smallest unit of a WLAN is an individual node or wireless STAtion (STA) [5]. Any device that accesses the wireless medium (WM) is effectively a STA, however some STAs are more specialised than general client nodes, for example an access point (AP), which contains a STA, but also performs other functions, discussed shortly. WLANs can form various network topologies, all covered by the primary classifications of [9]:

- **Basic Service Set (BSS)** — A single WLAN cell with two or more STA in wireless communication,
- **Independent Basic Service Set (IBSS)** — Ad-hoc WLANs with no infrastructure at all, and
- **Extended Service Set (ESS)** — Typical infrastructure WLAN deployments.

These basic concepts are presented here.

Basic Service Set (BSS) The BSS "is the basic building block for IEEE 802.11 WLANs" [5, p. 10]. A BSS consists of two or more wireless STAs that are able to communicate with each other via the wireless medium [9], as shown in Figure 2.1. This does not necessarily include an access point (AP) — discussed shortly — although, as an AP also contains a STA, it can. "If a STA moves out of its BSS, it can no longer directly communicate with other members of the BSS" [9, p. 10]. The services available in a BSS are those services that all STA must provide, the so-called Station Services (SS): authentication, deauthentication, confidentiality and medium access control (MAC) service data unit (MSDU) delivery [9, p. 15]. In other words, the BSS is only the radio coverage area and includes only the four basic Station Services - no distribution or other services.

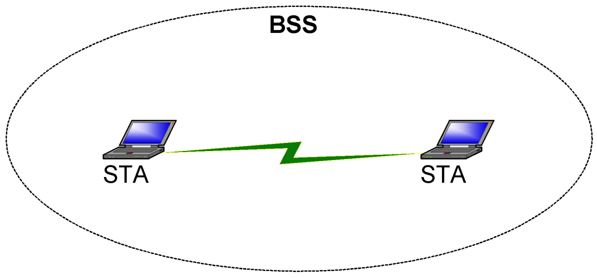


Figure 2.1: Basic Service Set (BSS)

The Independent BSS (IBSS) An IBSS, typically referred to as an ad hoc network, has no central control infrastructure, nor any fixed connection to wired infrastructure and is composed solely of STAs in communication with each other via the wireless medium, as shown in Figure 2.2. The services available in an IBSS are those SS that all STA must provide: authentication, deauthentication, confidentiality and MSDU delivery [9, p. 15].

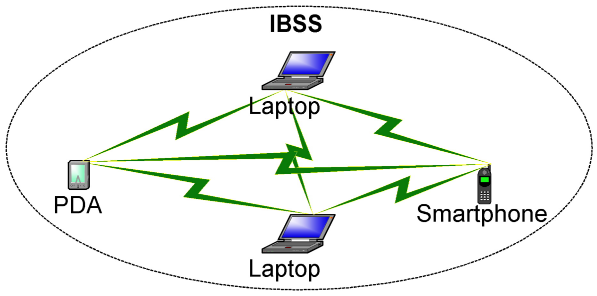


Figure 2.2: Independent Basic Service Set (IBSS)

Infrastructure Networks Where the STAs need services beyond their BSS, an infrastructure network is required. This consists of one or more wireless STAs controlled by an AP. The AP contains a STA that also provides services to allow the wireless clients access to a Distribution System (DS) [9, p. 15]. STAs must ***associate*** with the AP in order to access the DS, via the Distribution System Services (DSS): association, disassociation, reassociation, distribution (to another STA, possibly via another AP) and integration (with other networks, e.g. via a **portal** [9, p. 14] to wired infrastructure, possibly including proprietary intranets, proprietary internets or even the global Internet). All communication is through the AP, as shown in Figure 2.3.

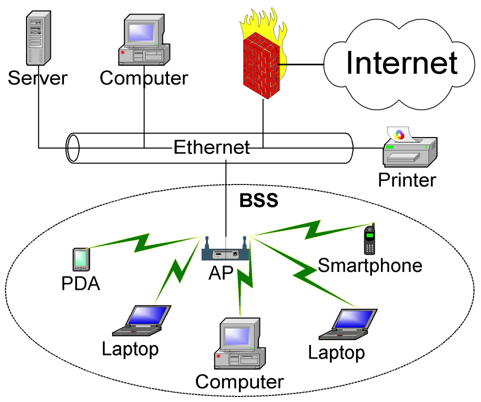


Figure 2.3: Infrastructure BSS

When a number of BSSs are connected together through some DS, they appear to the STAs as a single wireless network, or specifically an **Extended Service Set (ESS)** [66, p. 7], as shown in Figure 2.4. The ESS consists only of its component BSSs and does not include any other networks accessible through the DS. Thus, like a BSS, it is only the (combined or disparate) radio coverage areas. Stations should be able to roam from one BSS to another, within the one ESS, as if they were just one big BSS. However, as the original “standard” for packet forwarding from one AP to another in an ESS, *IEEE 802.11F* (See 2.4.2 for further details), was only a *Recommended Practice* — and has now been withdrawn, replaced by *IEEE Std 802.11r* — for older devices this is still mostly supported only by vendor-specific solutions and multi-vendor environments with older equipment may experience difficulties.

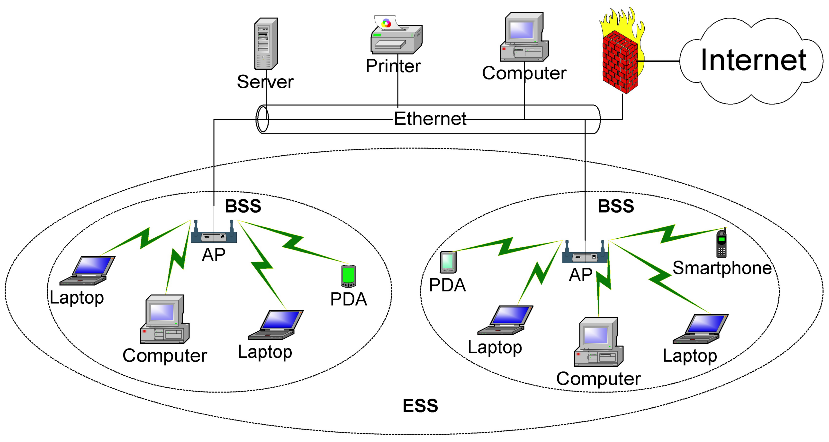


Figure 2.4: Extended Service Set (ESS)

IEEE 802.11 Operation

The physical layer (PHY) in IEEE 802.11, i.e. the wireless medium (WM) using electromagnetic radiation (EMR), is "fundamentally different" [9, p. 9] from wired media, such as twisted pairs of copper wire or coaxial cable. It has no clear boundaries; is unreliable; and is subject to the vagaries of mobility, dynamic environments, background EM noise, unrelated EM signals and atmospheric conditions.

IEEE 802.11 networks can have fixed, portable and mobile stations (STA). A portable STA can be moved from place to place but is only active in stationary locations. A mobile STA accesses the network while moving. IEEE 802.11 uses the IEEE 802 48-bit address space and so is compatible with the rest of the IEEE 802 LAN family [9].

IEEE 802.11 Authentication and Association

In infrastructure mode, a STA must first authenticate itself to an AP and then associate with the AP in order to access the DS, so that it can pass data via the AP to the DS — for distribution to other STAs at the same AP, other APs on the DS, or via a ***portal*** to a wired network [9, p. 14]. In practice, commercial "access points" are both an AP *and* a portal combined. A STA may only be associated with one AP at a time. An AP may have many associated STAs at the same time.

In ad hoc networks (Independent Basic Service Set (IBSS) mode) a node may authenticate itself to another STA. No association occurs as there is no DS. Each STA communicates directly with other STAs. A STA may be authenticated with many other STAs at the same time [9, p. 20].

IEEE 802.11 only provides link level authentication between STAs. It does not provide either end-to-end or user-to-user authentication [9, p. 20]. Authentication is either ***Open System*** authentication, where any STA can be authenticated without credentials, or ***Shared Key*** authentication, where the identity is validated by demonstrating the possession of the WEP encryption key [9].

Open System Authentication With ***Open System*** authentication, any STA can "authenticate" without credentials. That is, the identity is accepted without further authentication. However, if any form of encryption (WEP, WPA or WPA2) is being used, the STA will not be able to transmit or receive data without the correct keys [67].

Shared Key Authentication With ***Shared Key*** authentication, a challenge-response exchange is performed, using the shared WEP key to validate the STA. A challenge is sent to the authenticating STA, which encrypts the challenge using the shared WEP key and sends it back to the AP to verify the credential [67].

2.3 IEEE 802.11 WLAN Protocols

The list of IEEE 802.11 amendments, updates and enhancements keeps expanding and has now circled the alphabet into double-character suffixes. The following table (Table 2.2), updated from the author's 2005 paper [23], tries to give it some perspective.

2.3.1 IEEE 802.11a, b, g and "Super G"

In 1997, the Institute of Electrical and Electronics Engineers published *IEEE Std 802.11: Information technology— Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements— Part 11: Wireless LAN Medium Access Control (MAC) And Physical Layer (PHY) specifications* [5]. This detailed requirements for frequency hopping WLANs in the 2.4 GHz band (2400–2483.5 MHz) [68].[7] In 1999, the International Organization for Standardization (ISO[8]) adopted *ANSI/IEEE Std 802.11, 1999 Edition*, as *International Standard ISO/IEC 8802–11: 1999* [9].

This 2.4 GHz band (2400–2500 MHz) (limited to 2400–2483.5 MHz in the USA) is part of one of the international unlicensed Industrial, Scientific and Medical (ISM) bands, where communication services "must accept harmful interference" [69, p. 194] from ISM applications. With frequencies at this end of the microwave spectrum readily absorbed by water [70], signals are thus severely attenuated by water, and a major source of interference is the domestic microwave oven, along with other communications systems using this unlicensed band such as Bluetooth devices, some cordless telephones and the like [23].

[7]While this standard also detailed requirements for WLANs operating at infrared light frequencies, the scope of this work does not include infrared light WLANs.

[8]ISO is not an acronym, it is the organisation's short name (in any language). See http://www.iso.org/

Standards/Recommendations	**Status**
ISO/IEC 8802-11; ANSI/IEEE Std ***802.11****, 1999 edition.* Up to 2Mbps FHSS in the 2.4GHz band.	Existing Standard
ISO/IEC 8802-11:1999/Amd 1:2000; IEEE Std ***802.11a****–1999.* Up to 54Mbps OFDM in 5GHz band.	Existing Standard
IEEE Std 802.11b–1999 and *IEEE Std* ***802.11b****–1999/Cor 1–2001.* Up to 11Mbps DSSS in 2.4GHz band.	Existing Standard
IEEE 802.11 Task Group **C**. Information for bridge operations.	Work Complete
IEEE Std ***802.11d****–2001.* Specification for operation in additional regulatory domains (multi-country roaming).	Existing Standard
IEEE Std ***802.11e****–2005.* Initially both Quality of Service and security, standard now solely QoS.	Existing Standard
IEEE Std ***802.11F***[9]*–2003.* Recommended Practice for Multi-Vendor Access Point Interoperability.	Withdrawn
IEEE Std ***802.11g****–2003.* Up to 54Mbps in the 2.4GHz band. OFDM above 20Mbps, DSSS below 20Mbps.	Existing Standard
IEEE Std ***802.11h****–2003.* Spectrum and Transmit Power Management Extensions in the 5 GHz band in Europe.	Existing Standard
IEEE Std ***802.11i****–2004.* MAC Security Enhancements.	Existing Standard
IEEE Std ***802.11j****–2004.* 4.9 GHz-5 GHz Operation in Japan.	Existing Standard
IEEE Std ***802.11k****–2008.* Radio Resource Measurement.	Existing Standard
Letters 'l', 'o', 'q', 'x' and 'ab' not used.	
802.11m. Regular maintenance revisions of the base standard.	Currently 'TGmb'
802.11n. >100Mbps in 2.4GHz band.	Under Development
802.11p. Wireless Access for the Vehicular Environment.	Under Development
IEEE Std ***802.11r****–2008.* Fast BSS Transition (fast roaming).	Existing Standard
802.11s. Mesh networking.	Under Development
802.11T. Evaluation of 802.11 Wireless Performance.	Under Development
802.11u. Interworking with External Networks.	Under Development
802.11v. Wireless Network Management.	Under Development
802.11w. Protected Management Frames.	Under Development
IEEE Std ***802.11y****–2008.* 3650–3700 MHz Operation in USA.	Existing Standard
802.11z. Extensions to Direct Link Setup.	Under Development
802.11aa. Robust streaming of AV Transport Streams.	Under Development
802.11ac. Very High Throughput <6Ghz.	Under Development
802.11ad. Very High Throughput in 60 GHz.	Under Development

Table 2.2: IEEE 802.11 Standards

[9]A Recommended Practice is not an amendment, hence uses an upper case letter.

IEEE 802.11a

In December 1999, the first amendment, *IEEE Std 802.11a–1999* (*International Standard ISO/IEC 8802–11:1999/Amd.1:2000* [71]), added specifications to produce much higher data rates (6-54 Mbit/s) using orthogonal frequency division multiplexing (OFDM) in the 5.2 GHz band (5150–5350 MHz) and the 5.8 GHz band (5725–5825 MHz) [68]. Note that the 5.2 GHz band is directed at the USA-specific "unlicensed National Information Infrastructure (U-NII)" band, for which there is no equivalent purpose[10] in the *Australian Radiofrequency Spectrum Plan* [69] (nor for many other countries outside the USA). The 5.8 GHz band (but not the 5.2 GHz band) is also part of the international (including Australia) unlicensed ISM bands [23]. Also, the 5 GHz transmission frequency, although free of the interference of the 2.4 GHz band, considerably limits its range (at the same power) compared to 2.4 GHz.

There are 12 IEEE 802.11a channels in the 5.2 GHz and 5.8 GHz bands. The following table (Table 2.3) outlines the spectrum availability.

Frequency Band	**Channel Number**	**Centre Frequency**
USA only — UNII lower band	36	5.180 GHz
5.15-5.25 GHz	40	5.200 GHz
	44	5.220 GHz
	48	5.240 GHz
USA only — UNII middle band	52	5.260 GHz
5.25-5.35 GHz	56	5.280 GHz
	60	5.300 GHz
5.725-5.875 GHz	64	5.320 GHz
ISM 5.8 GHz band	149	5.745 GHz
5.725-5.875 GHz	153	5.765 GHz
(includes USA — UNII upper band	157	5.785 GHz
5.725-5.825 GHz)	161	5.805 GHz

Table 2.3: IEEE 802.11a Frequency Plan [73]

IEEE 802.11b

The second amendment, *IEEE Std 802.11b - 1999* [74], published in 2000, introduced specifications for higher data rates (5.5-11 Mbit/s) using direct sequence

[10]However, use of 802.11a WLANs is permitted under the provisions of the Australian Low Interference Potential Devices (LIPD) Class Licence [72].

spread spectrum (DSSS) in the original 2.4 GHz international ISM band [68]. The 2.4 GHz band (2400–2500 MHz) is shared with Bluetooth devices, cordless telephones and other devices. However, unlike Bluetooth's 79 channels, IEEE 802.11b's DSSS spreads the data frame over a 22 MHz wide channel. Because of this wide DSSS signal, there are effectively only three non-overlapping channels that can be used in any given location and so it is highly prone to interference from other devices with similar spectra, including domestic microwave ovens [23, 70].

There are 14 IEEE 802.11b DSSS channels, each 22 MHz wide [9, p. 218], with 13 spaced 5 MHz apart from 2.412 GHz to 2.472 GHz and channel 14 at 2.484 GHz [74, p. 42]. The diagram (Figure 2.5) and tables (Tables 2.4 and 2.5) outline the spectrum availability worldwide.

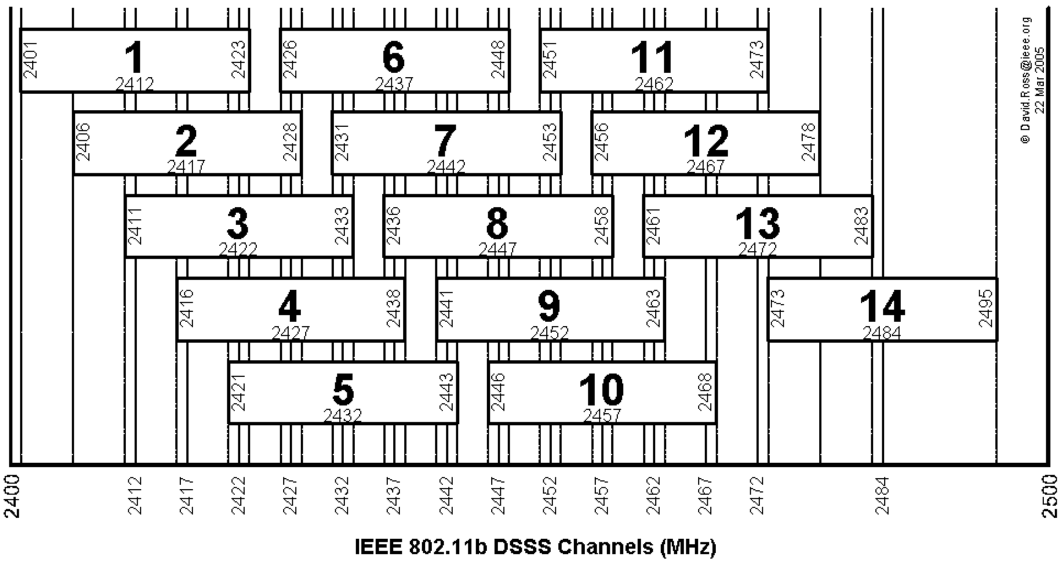


Figure 2.5: IEEE 802.11b Channel Layout

IEEE 802.11g

While IEEE 802.11a is significantly faster than IEEE 802.11b, the two are incompatible and IEEE 802.11b had, by far, the greatest market penetration [10, 11]. IEEE 802.11g [75] was a further addition, using IEEE 802.11a's Orthogonal Frequency Division Multiplexing over IEEE 802.11b's frequencies in the 2.4GHz ISM band. IEEE 802.11g can also fall back to IEEE 802.11b's DSSS operation and thus is fully backward compatible with IEEE 802.11b [23].

Channel Number	Centre Frequency	DSSS Range
1	2.412 GHz	2.401–2.423 GHz
2	2.417 GHz	2.406–2.428 GHz
3	2.422 GHz	2.411–2.433 GHz
4	2.427 GHz	2.416–2.438 GHz
5	2.432 GHz	2.421–2.443 GHz
6	2.437 GHz	2.426–2.448 GHz
7	2.442 GHz	2.431–2.453 GHz
8	2.447 GHz	2.436–2.458 GHz
9	2.452 GHz	2.441–2.463 GHz
10	2.457 GHz	2.446–2.468 GHz
11	2.462 GHz	2.451–2.473 GHz
12	2.467 GHz	2.456–2.478 GHz
13	2.472 GHz	2.461–2.483 GHz
14	2.484 GHz	2.473–2.495 GHz

Table 2.4: IEEE 802.11b Frequencies derived from [74, p. 42] ±11 MHz [9, p. 218]

Country	Channel Numbers
Australia & Europe	1 – 13
France	10 – 13
Japan	1 – 14
USA	1 – 11

Table 2.5: IEEE 802.11b Channels by Country

Super G "Super G" is not an IEEE standard. Vendors, such as Atheros, Netgear and D-Link use channel bonding, "which combines two 54 Mbps channels into one 108 Mbps [proprietary 'Super G'] link" [76]. "Super G" is thus incompatible with the standardised technologies, although such proprietary equipment will always offer the standard technologies as well. However, fall-back to the standard technologies may not necessarily be automatic [23].

2.3.2 IEEE 802.11n, "Pre-N" and "Draft-N"

IEEE Task Group N was chartered to define "high throughput" speeds for WLANs. The amendments are expected to provide payload throughputs up to at least 100 megabits per second (Mbps) to enable applications such as HDTV or voice over Wi-Fi (VoFi) [77].

IEEE 802.11n While 802.11g has a data rate of 54Mbps, it has an “actual throughput [of] about half that” [78]. Channel bonding two 54 Mbps channels into one 108 Mbps proprietary “Super G” link has limited practicality since there can be, at best, only 3 non-overlapping DSSS channels in the 2.4 GHz band. A new concept is now being finalised to dramatically increase throughput in the 2.4 GHz band. Four different consortia, MITMOT, TGn Sync, WWiSE and Qualcomm, had each put forward complete proposals for the new standard [79], to be IEEE 802.11n, based on Multiple Input Multiple Output (MIMO) technology.

MIMO Single Input Multiple Output (SIMO), Multiple Input Single Output (MISO) and Multiple Input Multiple Output (MIMO) provide better WLAN throughput than by using simple channel-bonding to increase bandwidth. SIMO, MISO and MIMO use multipath transmissions (echoes and reflections) and additional signal processing on both ends to provide an additional ‘spatial’ dimension to the transmission, to provide a dramatic increase in both throughput and range [80].

World Wide Spectrum Efficiency (WWiSE) The World Wide Spectrum Efficiency (WWiSE) consortium, including Airgo Networks, Bermai, Broadcom, Conexant, STMicroelectronics and Texas Instruments, has provided one of the proposals for the IEEE 802.11n standard [81]. It uses MIMO with at least two, up to four antennae at each STA, over the existing IEEE 802.11b/g 20 MHz channels, to achieve throughputs of 135 Mbps [82].

TGn Synch The TGn Synch consortium, consisting of Agere Systems, Atheros, Intel, Nokia and Sony, provided another of the proposals. The TGn Synch scheme also incorporates ODFM and MIMO, like WWiSE [76], as did the other two proposals, one from Qualcomm and the other from Mitsubishi and Motorola (MITMOT) [82]. “The largest difference is that the WWiSE proposal defines the same 20-megahertz-wide channel that current Wi-Fi standards use. The three other camps propose to bond two such channels into a new width of 40 MHz. Doing so nearly doubles data rate — from [WWiSE’s] 135 Mbps to TGn Synch’s 243 Mbps” [82].

Pre-N Although the IEEE 802.11n ratification has been shifted from November 2006 [82] to June 2010 [83], very early in the process, Belkin had already released laptop cards and a router that incorporated some of this technology, based on Airgo Networks' (WWiSE consortium) "True MIMO" technology, even though it may not be compatible with the eventual standard. Belkin originally called it "pre-11n", but that triggered concern in the Wi-Fi Alliance, the industry-based certifying body. So Belkin then simply called it "Pre-N"[11], although the reference was obvious [23].

Draft-N As the standards process progressed, by mid 2005, WWiSE, TGn Synch and MITMOT combined their proposals to develop the draft standard, which was accepted by TGn and went to Draft 1.0 ballot early in 2006. However. Draft 1.0 achieved less than half the vote. In January 2007 Draft 2.0 was released for procedural ballot, which was approved in February and immediately balloted for technical comment. In March, 2007 the technical vote closed and Draft 2.0 was approved to proceed [79].

By mid 2007 the Wi-Fi Alliance chose to use Draft 2.0 as the basis of its draft IEEE 802.11n certification program. This is the standard to which vendors currently produce "Draft-N" equipment, available to consumers through retail outlets, even though Draft 8.0 of the standard has now passed sponsor ballot and Draft 9.0 has been created to continue the standards process [79].

2.3.3 CSIRO's WLAN Patent

The Australian Commonwealth Scientific and Industrial Research Organization (CSIRO) holds US patent 5,487,069, issued on 23 January 1996 (filed on 23 November 1993) to provide high throughput wireless LAN in multipath environments [84], used in IEEE 802.11a/g and the emerging IEEE 802.11n equipment. The CSIRO has been in dispute with wireless vendors for many years, but has recently settled with Hewlett-Packard and is still in litigation with "Microsoft, Dell, Toshiba, Intel, Nintendo, Netgear, Belkin, D-Link, Asus, Buffalo Technology, 3com, Accton and SMC" [85].

[11]http://www.belkin.com/au/FactSheet/Belkin_PreN_Fact_Sheet.pdf

2.4 Other IEEE 802.11 Recent Standards

This section, as part of the current literature reviewed at the beginning of this research, briefly describes those amendments that had been ratified at that time. Note that subsequent amendments are described later.

2.4.1 IEEE Std 802.11d–2001

The “Specification for operation in additional regulatory domains” [86] for multi-country roaming was finalised in 2001.

2.4.2 IEEE 802.11F

“IEEE Trial-Use Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation” [87] was an inter-access point protocol to support roaming STAs.

The capital ‘F’ designated a ‘recommended practice’ [88] — a stand-alone document and not an amendment to the IEEE Std 802.11 itself which would use lower-case postfixes.

IEEE 802.11F allowed for roaming between APs on the same network segment, but this often failed when crossing network segments, particularly for VoIP applications [89]. “Handoff for voice needs to be no more than 20 milliseconds” [89, (quoting Paul Congdon, Hewlett-Packard)]. With this limitation and the expenses of IEEE 802.11i and RADIUS, VoFi systems used no encryption or the insecure WEP in order to ensure the handoff reauthentication time remained under 20-milliseconds [89]. This is now obsoleted by the work of Task Group R.

2.5 Other 802.11 Emerging Standards

This section briefly describes those amendments that were under development at the beginning of this research. Some of these have been completed and are described with the advances in the field in Chapter 3. Others are still ongoing.

2.5.1 IEEE 802.11e

Task Group E was developing Quality of Service (QoS) provisions for WLANs, which it completed in 2005, drawing on the additional management frames provided by Task Group H, as described in next chapter [section 3.1.3].

2.5.2 IEEE 802.11h

Task Group H was looking at the spectrum and power management for Europe, which it completed in 2003, providing new management frames also used by subsequent advances and directly related to this work, as described in next chapter [section 3.1.1].

2.5.3 IEEE 802.11i

Task Group I was working on the enhanced MAC security provisions, which it completed in 2004 and forms the bulk of the subject matter for the advances in the next chapter [section 3.2].

2.5.4 IEEE 802.11j

Task Group J was adding regulatory requirements for use in Japan, which it completed in 2004, as described in next chapter [section 3.1.2].

2.5.5 IEEE 802.11k

Task Group K was investigating radio management information messages, which it completed in 2008, drawing on the additional management frames provided by Task Group H, as described in next chapter [section 3.1.4].

2.5.6 IEEE 802.11p

Task Group P is investigating "Wireless Access in the Vehicular Environment (WAVE)".

2.5.7 IEEE 802.11r

Task Group R was working on fast handoff between BSS APs to support streaming technologies such as VoFi. This completed in 2008, again drawing on the additional management frames provided by Task Group H, along with the radio management information messages from Task Group K, as described in next chapter [section 3.1.5].

2.5.8 IEEE 802.11s

Task Group S (TGs) is investigating wireless mesh protocols for access points, allowing for multi-hop transmissions via mesh routing protocols. Although the IEEE 802.11s standard is not ready yet, a number of vendors offer mesh networking products that allow enterprises to deploy WLAN APs in hard-to-cable locations or allow frequent movement of APs. Mesh APs require power, but the only data cabling is on the edge of the mesh [90]. Current solutions are unique to each vendor, whose hardware is not interoperable with others. Also, both back-haul and user access will contend for bandwidth, degrading performance.

2.5.9 IEEE 802.11v

Increasing uptake of WLAN will make management of enterprise-wide APs more difficult. There is a growing need for remote wireless network management of configurations and software. Typically, this is performed via the SNMP, but currently there are no standard MIBs for WLAN-specific parameters, so this varies from vendor to vendor. Currently, enterprises have to standardise on one vendor's hardware or else use a third-party tool that attempts to manage hardware from multiple vendors.

2.5.10 IEEE 802.11w

Task Group W (TGw) is investigating Protected Management Frames, as discussed in Chapter 4, section 4.7.

2.6 IEEE 802.11 WLAN Security

This section gives the details of the various security provisions and attempts to provide security for the original IEEE 802.11 WLANs and the various issues with those provisions subsequently discovered in deployments under the original standard.

Sections of this work were also presented by the author in "*Securely Deploying IEEE 802.11 WLANs*" [24], at the *AusCERT Asia Pacific Information Technology Security Conference (AusCERT2007)*, Gold Coast, Australia, in May 2007.

Wired networks at least provide a measure of physical security in that they, even if using broadcast paradigms, limit the dissemination of the signal to the wired medium itself and the attached stations (STA), for all but exceptionally well-resourced adversaries. Wired networks are also typically under the physical control of the network owner and are secured in machine rooms or located in ducts and conduits or inside walls, hidden above ceilings or buried under the ground.

Whereas, the wireless medium (WM) permeates everywhere, reaching everyone, friend or foe (without the adjunct of special purpose buildings or environments to gaol electromagnetic emissions) and wireless networks generally broadcast their signals onmidirectionally and unconstrained, limited only by the sensitivity of a receiver to extract an attenuated signal from the background noise. These characteristics jeopardise **confidentiality** by providing information, not only in the content of the signal, the data in the message body and the identity information in the message headers, but in the signal itself, its strength and location [24].

The characteristics of the medium make WLANs highly susceptible to attacks on the **availability** of the services (DoS attacks), through either directional or omnidirectional jamming of the medium itself, or any one particular channel thereof [91], or the easy directional or omnidirectional insertion of unprotected, unauthenticated management frames [92] to deauthenticate or disassociate a STA from its controlling AP. These latter attacks can be used in isolation or as the initial part of a more sophisticated attack to interrupt communications as a prelude to various masquerading attacks.

The lack of inherent confidentiality and weak protection of availability, coupled with the intrinsic ability for easy injection of traffic into the medium by any STA, anywhere, with sufficient transmitting power and appropriate antennae, also threatens WLAN information **integrity**. Hostile STAs may masquerade as legitimate STAs to an AP or as the legitimate AP to an unsuspecting STA or even both at the same time as a man-in-the-middle, completely controlling the communications between the legitimate STA and AP [24].

The following details the information security features (or features used in an attempt to support information security) for the original IEEE 802.11 WLAN standards. These set a baseline against which the current state of WLAN security (Chapter 3) and the emerging work and remaining issues (Chapter 4) can be compared.

2.6.1 Service Set Identifier (SSID)

In WLANs, the Service Set Identifier (SSID) acts as a WLAN name, in a similar manner to the '*community*' string in the ubiquitous Simple Network Management Protocol (SNMP). "Thus all devices trying to connect to a particular established WLAN must be configured with the same SSID" [33, p. 45]. Many implementations of WLANs attempt to use the SSID as a password, a purpose for which it is not designed and thus is not secure. Because the SSID is included in beacon frames by default, it provides no security. An intruder can get the SSID from the beacon frames and act as a legitimate node. If the SSID is suppressed in the beacons (a 'hidden SSID'), an intruder can simply watch for probe response frames in legitimate traffic and extract the SSID from there. If there are no probe response frames readily forthcoming, an intruder can simply send a forged disassociate packet to force a STA to reassociate and thus obtain the SSID.

2.6.2 MAC Address Filters

An AP can be configured to only accept association and connection requests from particular MAC addresses [33, p. 46]. Using the MAC address for authentication has a number of issues. Firstly, the MAC address is easily spoofed on any modern network interface. Moreover, the MAC address, IP address, default gateway and other details for legitimate clients is easily sniffed from the WM [93]. The MAC address is available in clear text even under the latest security protocols. Also, this typically only "establishes the identity of the [network interface], not its human user, so an attacker [illegitimately using] a laptop with a registered MAC address will appear to the network as a legitimate user" [33, p. 47]. In practical applications this provides no security.

2.6.3 Denial of Service (DoS)

DoS can be achieved through either directional or omnidirectional jamming of the medium itself, such as interference produced by some sort of broadband emitter or signal generator, or any one particular channel thereof [91], or the insertion of spoofed management or control frames.

DoS Attacks on Management Frames and EAP Frames

There are a number of DoS attacks using Management frames in IEEE 802.11 networks. The simplest of these is a fake Disassociate message forcing a STA to renegotiate association with the AP or a fake Deauthentication message forcing the STA all the way back to the unauthenticated state, to renegotiate authentication and association. Such messages can also be sent to the broadcast address to reset all the STAs in the network at once. They can also be repeated over and over to maintain complete DoS in a ‘Deauth flood’.

Other ways to achieve a DoS generally involve overloading the access point with such things as a flood of Probe Requests or a flood of Authentication Requests or Association Requests. These can be performed with MAC address spoofing to simulate a large numbers of STAs attempting to join the network.

Other device-specific attacks can affect the infrastructure supporting even Robust Security Networks (RSN). A number of DoS attacks can be effected on certain IEEE 802.1X infrastructure via malformed EAP frames. Vulnerable devices may suffer only temporary DoS (process crash with automated restart), or memory leaks until the process eventually crashes or even the possibility of executing arbitrary code on the authentication server. These are not vulnerabilities in the WLAN, but in the the supporting infrastructure exposed by the WLAN.

DoS Attacks on Control Frames

DoS can also be achieved using spoofed control frames. This is easily performed by flooding a network with fake CTS frames so that STAs believe the medium is reserved for a ‘hidden node’ for which they have not seen the RTS frames. These can be transmitted with sufficient frequency to render the channel unusable.

An alternative to this is to send fake RTS packets to the access point itself, so that it creates the CTS frames affecting every STA within range — every STA in the network.

2.6.4 Rogue Access Points

Rogue APs are unauthorised APs providing (alleged) DSS. These may be unauthorised APs attached to a corporate LAN by a legitimate user seeking wireless flexibility, often also providing unfettered access to the corporate network for any adversaries, corporate or otherwise. Alternately, it may be an adversary's AP pretending to be a legitimate AP, such as for a wireless Hot Spot, to defraud legitimate clients into providing their authentication credentials [33].

2.6.5 Hijack Attack

Far less likely is the abuse of routing protocols to allow eavesdropping on victim out-of-range of the attacker by detouring the traffic through corrupted nodes within the transmission range of both victim and attacker [94].

2.6.6 Wired Equivalent Privacy (WEP)

The Wired Equivalent Privacy (WEP) cryptographic confidentiality algorithm was designed to be "subjectively equivalent to the confidentiality of a wired local area network" [5, p. 6]. WEP is based on the RC4 cipher and a secret key that is shared between all of the nodes in the wireless LAN. It was intended to provide security properties similar to that of wired networks, however it suffers from a number of issues [33], as follow.

WEP One-way Authentication

WEP provides only client authentication and does not authenticate the AP. This provide the opportunity for a rogue AP to defraud a legitimate client into associating with it and passing those credentials on to the legitimate AP to realise a MITM attack.

Static WEP Keys

A major problem with WEP keys is the lack of key management provisions within the protocols, so that the same key is used by all stations and requires manual processes to be performed at each of the STA for any key rotation, which is therefore infrequently, if ever, performed. This allows large amounts of ciphertext for the same key to be gathered by any adversary — making cipher-text only attacks considerably easier and faster.

WEP Key Key Size

The USA's export restrictions on cryptographic material that existed during the development of the original protocols left WEP with an inadequate 40-bit key coupled with a 24-bit initialisation vector to create a 64-bit RC4 cipher key. Many parties blamed 40-bit RC4 keys for WEP's weakness and recommended "using 104 or 128-bit RC4 keys instead" [33, p. 47].

However, using the larger key size did little to dissuade adversaries, who simply required more packets for the same unchanging key to effect an attack. Jesse Walker's submission [95] to the IEEE 802.11 working group, identifies an inherent vulnerability in WEP's initialization vector irrespective of key size [33].

WEP is easily cracked (defeated) using commonly available software, such as *WEPCrack* [96], *AirSnort* [97], *Kismet* [98], *aircrack*[12] [99], *Aircrack-ng* [100] and *WepLab* [101], among others.

Commonly Available Wireless Attack Tools

Some of the more common tools used to investigate wireless networks are included here.

Kismet Kismet is a passive (optionally active) wireless network detector and promiscuous packet capture system for Unix-like operating systems. It can be completely passive, purely listening to traffic to determine the networks in range [98] or it may also be configured to use active probes.

Wireshark Ethereal, now developed as Wireshark, is the best-known open-source passive software protocol analyzer that has been ported to most operating systems. Originated by Gerald Combs, it has an active contributing community and includes analysis of IEEE 802.11 networks [102].

NetStumbler NetStumbler (and MiniStumbler) are tools for Windows (and Windows CE) that detect IEEE 802.11 WLANs [103] and provide all of the avail-

[12]Christophe Devine does not capitalise *aircrack*, however Thomas d'Otreppe does capitalise *Aircrack-ng*.

able network details from the traffic present. They also use active methods, probing for available networks.

AirSnort AirSnort recovers WEP encryption keys by passively monitoring transmissions, however active methods can also be employed to speed this process. AirSnort uses the weakness described in Fluhrer, Mantin and Shamir [104]. "AirSnort requires approximately 5-10 million encrypted packets to be gathered. Once enough packets have been gathered, AirSnort can guess the encryption password in under a second" [97].

Airsnarf Airsnarf is a rogue wireless AP setup utility for Linux [105]. It accesses and copies a legitimate wireless hot spot sign-in page. Then it broadcasts a deauthenticate message that disconnects any nearby hot spot users from the legitimate AP and it presents itself as the legitimate AP [106].

AirJack AirJack provides low-level device drivers so that raw IEEE 802.11a, IEEE 802.11b or IEEE 802.11g frames can be crafted and injected into a wireless network, "providing the ability to perform a DoS attack or to actively determine the ESSID for a closed network and establish a man-in-the-middle attack" [107].

The KoreK Attacks

One of the primary weaknesses in WEP was fixed by vendors filtering the weak IVs. The Fluhrer-Mantin-Shamir attack [104] requires many millions of encrypted packets, so as to gather sufficient packets with weak initialisation vectors (IVs). Without these weak IVs and with regular key changes, WEP was, for a time, secure enough for the majority of applications [23].

Then in August 2004, "a hacker named KoreK posted new WEP statistical cryptanalysis attack code to the NetStumbler[13] forums" [108]. This was incorporated into a number of new tools, including ***aircrack*** [99] and ***WepLab*** [101]. These attacks do not require millions of packets to crack a WEP key and the number of weak IVs does not matter, only the total number of unique IVs captured [11,23]. "A key can often be cracked with hundreds of thousands of packets, rather than [the] millions" [108] previously required.

[13]http://www.netstumbler.com/

aircrack Christophe Devine developed aircrack, which implements the KoreK attacks as well as the improved Fluhrer-Mantin-Shamir attack from dwepcrack [109] by David 'h1kari' Hulton [108].

WepLab Jose Ignacio Sanchez developed WepLab, which also implements the KoreK attacks. WepLab also provides brute force and dictionary cracking attacks (as does aircrack) [101].

2.6.7 Initial Attempts at Improving Security

With these crippling vulnerabilities in the IEEE 802.11 protocols, alternate protection methods had to be deployed to provide an acceptable level of security in the existing WLANs. Prior to the release of the IEEE 802.11i amendment, wireless networks were best treated as untrusted networks, like the public Internet. Additional security mechanisms, such virtual private network (VPN) tunnelling, including those using the *Security Architecture for the Internet Protocol* (IPsec) [110], Secure Sockets Layer (SSL) encapsulation and various vendor implementations of the *IEEE Std 802.1X* [111] port-based network access control (NAC) framework [67] were often used to help mitigate the atmosphere of insecurity in these WLANs. The need for such wrappers and workarounds in WLAN deployments indicated the lack of acceptably secure wireless networking protocols and hardware. Three of these alternative solutions for WLANs presented by Zahur and Yang [33] were: Secure Sockets Layer (SSL), Virtual Private Networks (VPN) and Cisco's Lightweight EAP (LEAP).

Secure Sockets Layer (SSL)

The Secure Sockets Layer (SSL) protocol [112] was originally developed by Netscape Communications Corporation[14] as a "bubble-in-the-stack", inserted as a sublayer between the the Transport Layer and the higher layers of the protocol stack, to provide a "Berkeley-Sockets[15]-like" application programming interface (API) supporting secured inter-process communications. SSL Version 2 [113, 114] was released in 1994, in a draft RFC submitted to the W3O working group on security. SSL Version 3 was released in late 1995 [115] and as an Internet-Draft [112]

[14]Originally Mosaic Communications Corporation, then Netscape Communications Corporation, later Netscape Communications, later purchased by AOL LLC and later dissolved.

[15]The University of California at Berkeley's inter-process communication API for UNIX.

by the Transport Layer Security Working Group in late 1996.

In this manner, higher-layer protocols may be able to utilise SSL connections to compensate for these initial WLAN security shortfalls. While SSL is available for a variety of applications, such as HTTP or SMTP over SSL, there are still many applications that cannot support it. However, the increasing popularity of the deployment and use of SSL-based VPNs would help support this functionality.

SSL relies upon digital certificates and a Public Key Infrastructure to authenticate the server-side only — the client is typically unauthenticated. This pushes the burden of authentication of the STA onto the AP or onto the associated networks behind AP.

Also, in the nature of this SSL certificate authentication, some applications may accept any valid certificate under one of the many in-built root Certificate Authorities (CA) found in common operating systems. This means the client of the application at the STA must be diligent in verifying that the identity presented by the application's server exactly matches the identity it was expecting.

Virtual Private Networks (VPN)

VPNs are available in a multitude of proprietary vendor configurations, as well as in a number of open source implementations, the most popular of the latter being the IPsec [110] protocol suite providing security at the IP layer. VPNs are frequently used to provide secure communications for the higher layers of the communication protocols over otherwise insecure wired networks such as the Internet. This protects the actual data being transferred, however leaves the devices themselves and the communications channels still exposed to attack via known, unpatched, or zero-day attacks. VPNs provide the security services of authentication, confidentiality and integrity [33]:

- Authentication: Depending on the vendor, or the configuration, VPNs can be anonymous, unilaterally authenticated or bilaterally (mutually) authenticated. To provide adequate security, the VPN and should be used to authenticate both the wireless user and the network the user is connecting to. As both parties can be authenticated at the start, the VPN can thus provide authentication of the source or of the recipient or both.

- Confidentiality: The VPN will tunnel all traffic using secure protocols to encrypt all data and protect its confidentiality while traversing the network.

- Integrity: As part of this tunnelling process VPNs will normally also provide data integrity, via encrypted message integrity checks (MIC), of the tunnel payloads wherever appropriate encryption has been used.

Cisco's Lightweight EAP (LEAP)

Cisco's LEAP was Cisco's interim solution for wireless security and used a proprietary Extensible Authentication Protocol (EAP) method. The EAP framework was designed to support various authentication methods in peer-to-peer associations, such as the Point-to-Point Protocol (PPP). LEAP was used to provide mutual authentication between a client and an Authentication Server (AS) via the AP and is carried within the Remote Authentication Dial In User Service (RADIUS) protocol on the wired LAN.

LEAP was to provide mutual authentication and WEP enhancements such as a MIC and per packet keying [116]. However, LEAP used a modified version of Microsoft's MS-CHAP and the user credentials were easily compromised. The *Cisco Secure Access Control Server (ACS) User Guide* [117] now recommends disabling LEAP in favour of the stronger protocols, such as EAP-TLS, PEAP or EAP-FAST.

The use of such EAP methods led to a renewed popularity of the use of the IEEE 802.1X port-based network access control (NAC).

IEEE 802.1X Authentication Protocol

In IEEE 802.1X there are three entities, the supplicant, an authenticator and an authentication server [111]. The client STA is the supplicant, communicating with the AP as an authenticator, however actual authentication is performed with the authentication server (AS). IEEE 802.1X provides two main features [66] when deployed in the wireless environment: logical ports and key management.

Logical Ports Wireless STAs are not connected to physical ports like their wired counterparts, but are associated with logical ports on the AP. In associating with the AP, the combination of the STA's and the AP's IEEE (MAC) addresses establish a logical port for the connection to the wireless device. This logical port then acts as a destination address in "EAP over LAN" (EAPOL) protocol exchanges [66]. In an ‘unauthorised’ state, the port allows only the Dynamic Host Configuration Protocol (DHCP) and EAP traffic to pass through. Once ‘authorised’, as part of the association process, the port has the normal access to the DS and other portal functions.

Key Management The other principal feature of IEEE 802.1X is the key management. Session keys are established with the handshake at the beginning of the connection on a per session basis using EAPOL-Key message and can also be rotated as required, over particular periods of time during the session.

IEEE 802.1X Vulnerabilities

Arunesh Mishra and William Arbaugh from University of Maryland published design flaws and the resulting vulnerabilities in certain configurations of IEEE 802.1X with WLANs, including “Absence of Mutual Authentication” [118, p. 7] and “Session Hijacking” [118, p. 8]. As with all technologies, various implementations or insecure configurations can expose vulnerabilities.

In the case of IEEE 802.1X, it is not the framework itself, but poor choice of EAP methods that can expose the network. This is particularly true of EAP methods that do not involve mutual authentication or rely on weak cryptographic components, such as EAP-MD5. These issues are discussed in greater detail in the next chapter, Chapter 3, as part of the advances announced during the course in of this work.

Details of the MAC security enhancements and the emerging Management Frame protection amendments and other recent and emerging standards are also discussed in Chapter 3.

Chapter 3

State of the WLAN Security Art

This chapter expands on the background information in the previous chapter and describes the current state of the art for IEEE 802.11 wireless network security. This chapter details the various advances in the field that occurred during the conduct of this research, including the recent amendments for the MAC security enhancements and how they are implemented in Wi-Fi Protected Access (WPA) and WPA2. It discusses and compares the differences between WEP, WPA, WPA2, *IEEE Std 802.11i*, the Robust Security Network (RSN) and the Transition Security Network (TSN).

This chapter also identifies a number of remaining security issues in today's WLANs that are resolved by this research, some of which are also the subject of emerging work discussed in Chapter 4, but unlikely to be completely resolved by that work — as well as some issues that remain unresolved by any current or emerging activities.

Most of the material here was also presented by the author in "*Securely Deploying IEEE 802.11 WLANs*" [24], at the *AusCERT Asia Pacific Information Technology Security Conference (AusCERT2007)*, Gold Coast, Australia, in May 2007. Some sections of this material were also presented in an Australian Academic and Research Network (AARNet) 'Ozeconference' titled "A Review of Actual IEEE 802.11 Deployment Practices" [11] in May 2008.

Section 3.1 details all of the amendments to the *IEEE Std 802.11* released during the course of this research, each of which impacts, either directly or indirectly, on the current state of WLAN security, including the various advances introducing even more sensitive management data to traverse the WM unprotected. Section 3.2 describes the MAC security enhancements, which only protect the payload of data frames, leaving frame headers, control frames and management frames unprotected in WLANs. Section 3.3 then details the requirements for an RSN and RSNAs, the implications of a TSN, what the WPA certification and the WPA2 certification actually provides, along with the conditions where WPA2 actually implements a TSN instead of an RSN. Finally, section 3.4 discusses the results of a series of tests that were presented in the author's AusCERT2007 paper [24], mentioned above, demonstrating the vulnerabilities from weak configurations of WPA and WPA2 WLANs.

This then completes the current state of related work in the field and forms the basis of the new research detailed in the following chapters.

3.1 Advances During This Work

Various wireless networking technologies, including higher throughput approaches, Quality of Service (QoS) provisions, wireless network management, wireless roaming handoffs and WLAN security in particular, were the subject of significant advances during the course of this research.

With these advances, WLANs gain far more utility for use in mainstream commercial and even government applications. The advances in QoS and radio management, combined with the new roaming fast BSS transition standards make VoFi a practical possibility for commercial environments, where confidentiality is an issue.

Manufacturers are already producing dual-mode cellular/VoFi handsets using the recent IEEE 802.11 amendments (including e, h, k and r) which allow these handsets to utilise WLANs for high-speed Internet and VoIP telephone calls. VoFi is expected to be "a core driver of the enterprise WLAN market" [80].

The first of these advances was the IEEE Std 802.11h-2003 Spectrum and Transmit Power Management Extensions for Europe [119], followed by the IEEE Std 802.11i-2004 MAC Security Enhancements [12] and the IEEE Std 802.11j-2004 5 GHz Operation in Japan [120], then the IEEE Std 802.11e-2005 Quality of Service [121], a roll-up of all the amendments into IEEE Std 802.11-2007 [122] and then the IEEE Std 802.11k-2008 Radio Resource Measurement [123] and IEEE Std 802.11r-2008 Fast Roaming [124] amendments. These advances and the current state of the art of WLAN security are detailed in the following sections, in chronological order except for the IEEE Std 802.11i-2004 MAC Security Enhancements (lastly) forming the bulk of the detail supporting the information presented in this Chapter.

3.1.1 IEEE Std 802.11h-2003

“Spectrum and Transmit Power Management Extensions in the 5 GHz band in Europe” [119] was originally designed to handle frequency selection and transmission power control via management messages between APs and STAs to minimise interference with radar and satellite communications in the 5 GHz bands in Europe, however these amendments have implications in many more regulatory domains and can be used to improve WLAN efficiency. The devices “select another channel and adjust power output” [88] as required.

In order to achieve this, *IEEE Std 802.11h* defines a series of new management frames called *Action* Management Frames, that are used to transmit spectrum management directives. These Action Management Frames are also carried into the *IEEE Std 802.11e*, *IEEE Std 802.11k* and *IEEE Std 802.11r* work.

3.1.2 IEEE Std 802.11j-2004

This amendment modified the 802.11 MAC and 802.11a PHY for the 4.9–5.0 GHz spectrum to meet Japanese regulatory requirements.

3.1.3 IEEE Std 802.11e-2005

The ever expanding suite of applications being delivered over WLAN has lead to traffic management issues as the bandwidth becomes saturated and the IEEE 802.11 contention-based services are not suited to streaming applications such as VoIP [89]. Task Group E was initially investigating both quality of service and security. This was then split with IEEE 802.11e for QoS and IEEE 802.11i for security [23]. IEEE Std 802.11e [121] extends the use of IEEE Std 802.11h’s Action Management Frames to handle components of the QoS messaging, making these Management frames critical to the successful deployment of the technology.

3.1.4 IEEE Std 802.11k-2008

IEEE Std 802.11k provides “Radio Resource Measurement of Wireless LANs” [123]. It defines a set of management messages to query information from the network STAs, such as an AP requesting that a STA report all beacons it can see on a particular channel in order to make traffic decisions about switching channels and the like [125]. Access points may also query STAs for lists of hidden nodes or for various radio measurements that are available from the STA’s network interface drivers [125]. This standard can also support the use of IEEE 802.11r to enhance fast transition between the APs.

This is all achieved through the extension of the use of the Action Management Frames of IEEE 802.11h, that are used extensively in this query and response process, pushing more and more network management information into the actual IEEE 802.11 management frames.

3.1.5 IEEE Std 802.11r-2008

Task Group R (TGr) is looking at fast roaming between APs — “handing off clients quickly from one access point to another with their authentication and security policies intact becomes critical when clients are moving, such as with VoIP calls made with hand-held WLAN phones” [88] or dual-mode GSM/VoFi phones.

IEEE Std 802.11r allows for fast BSS transition, that is, fast handoff from one AP to another to maintain mobile streaming services such as VoIP. Prior to this, the many steps to handoff — a passive or active scan for APs in the area, exchange authentication messages with the new AP, exchange reassociation messages to establish a connection at the new AP, performing the IEEE 802.1X EAP negotiations, pairwise master key generation and pairwise transient key derivation and establishing quality of service requirements — made streaming services impractical under IEEE 802.11i. IEEE 802.11r avoids new key exchanges during handoff within a “mobility domain” [124, p. 2] and carries all resource and quality of service messages with the IEEE 802.11 authentication and reassociation messages and leverages any advantages of IEEE 802.11k as well.

3.2 IEEE Std 802.11i-2004 — MAC Security

With the original insecurities of WLANs being well known and easily exploited [33, 43, 95, 106, 108, 126], both industry and standards have moved to respond to the issues [24]. Task Group I was separated from the original Task Group E, looking at both quality of service and security, so that the security issues were the single focus for IEEE 802.11i.

The *IEEE Std 802.11i* [12] amendment provides an effective and more secure replacement for WEP, providing two new confidentiality protocols, Temporal Key Integrity Protocol (TKIP) for the legacy RC4 hardware and the Advanced Encryption Standard (AES) in CCM[1] Protocol (CCMP), enabling authentication and authorisation via the the IEEE 802.1X port-based Network Access Control (NAC), providing key distribution via the Extensible Authentication Protocol over LAN (EAPOL) key exchange [127].

IEEE 802.11i resolved the problems with WEP and the confidentiality and integrity (including authentication) of all data frames [24]. However, control and management frames remain unsecured, as are the link layer headers, and DoS attacks on the availability of the services still exist, along with some new DoS attacks specific to new elements in IEEE 802.11i [92]. Even if using a RSN, the lack of protection for management or control frames still provide multiple attack vectors into these networks.

Where data frames are protected by the IEEE 802.11i amendments, an RSN, using CCMP, requires new hardware to operate and so deployments with provisions for legacy hardware will typically fall back to using the TKIP or even to a TSN that allows Pre-Robust Security Network Associations (Pre-RSNA) [24].

Temporal Key Integrity Protocol (TKIP) TKIP was designed to improve the security of legacy products that had implemented WEP, but were incapable of supporting the Advanced Encryption Standard (AES). It uses a Message Integrity Code (MIC) called Michael and provides per-frame keying [127].

Counter-Mode/CBC-MAC Protocol (CCMP) CCMP is for new hardware and is stronger than TKIP. It uses AES in counter mode with 128-bit blocks and a 128-bit key. “For authentication and integrity, CCMP uses Cipher

[1]CTR (CounTeR mode) with CBC-MAC (cipher-block chaining (CBC) with message authentication code (MAC)) [12, p. 5].

Block Chaining Message Authentication Code (CBC-MAC)" [127]. The MAC is 8 octets, the nonce is 48 bits (6 octets) and two extra bytes of 802.11 header make a total of 16 extra bytes per packet. CCMP protects some fields that are not encrypted, the additional authentication data (AAD), which include the source and destination [127]. However, these only provided integrity protection in detecting modification or damage and do not provide protection for either availability or confidentiality.

IEEE 802.1X use in an IEEE 802.11i system In IEEE 802.1X there are three different types of entities, a supplicant, an authenticator and an authentication server [111]. The authenticator does not actually authenticate the supplicant, it merely enforces authentication. The authenticator communicates with the supplicant, which it does not allow access to the network, other than DHCP[2] and EAP, and passes the supplicant's credentials back to the authentication server. The authentication server authenticates the supplicant and passes the result to the authenticator, which then permits or denies access to the network [24].

In IEEE 802.11i, the Access Point is the Authenticator only, with a typically separate authentication server, typically a Remote Authentication Dial-In User Service (RADIUS) server, such as a Cisco Access Control Server (ACS), or an RSA ACE/Server, Microsoft IAS server, or the like, and the client NIC is the Supplicant. Thus, the client NIC (supplicant) authenticates with the authentication server via the AP (authenticator). The IEEE 802.1X protocol is used between the client NIC and the AP. The protocol between the AP and authentication server is not specified in either IEEE 802.1X or IEEE 802.11i, but this is typically a version of the RADIUS protocol [127].

In IEEE 802.1X, the 'uncontrolled port' is used for authentication traffic between the client NIC (supplicant) and the authentication server. Once the authentication server authenticates the client NIC (supplicant), the authentication server informs the AP (authenticator) and passes the necessary key material to the AP (authenticator), so that both the STA and AP have the key material and the AP allows access through the 'controlled port' [127].

Protocol Negotiation The *group ciphersuite* is the protocol used to encrypt broadcast traffic. The *pairwise ciphersuite* is a list of protocols allowed for unicast

[2]Dynamic Host Configuration Protocol

traffic. The *authentication and key management suite* advises whether preshared key (PSK) mode or IEEE 802.1X is available [127].

Key Exchanges The IEEE 802.11i pairwise master key (PMK) comes from the authentication server or a preshared password mapped into a PMK. This is fed into a pseudorandom function, along with the STA's and AP's MAC addresses, a nonce from the STA and a nonce from the AP, to create the pairwise transient key (PTK). The PTK gets divided into five keys: the EAPOL Key Encryption Key (KEK), the EAPOL Key Confirmation Key (KCK), the Temporal Key (TK), the Tx MIC Authenticator Key and the Rx MIC Authenticator Key. The KCK is used for data origin authenticity. The KEK is used for key confidentiality. The TK is used for data confidentiality [127].

The random GMK, the AP's MAC address and a nonce are fed into a pseudorandom function to create the group temporal key (GTK) [127].

The 4-way Handshake [127]

1. The authenticator sends a nonce, *ANonce*, to the supplicant.
2. The supplicant uses this and its own nonce, *SNonce*, to calculate the *PTK*. The supplicant sends its *SNonce* and the security parameters it used to the authenticator, authenticated using the *KCK*. The authenticator then verifies that these are valid.
3. The authenticator sends its security parameters and the *GTK* encrypted with the *KEK*, all authenticated using the *KCK*, to the supplicant, which allows the supplicant to verify the authenticator's information.
4. The final acknowledgement activates encryption.

Group Key Handshake Only to update the *GTK*, since 4-way handshake includes the *GTK*.

1. The AP sends the *GTK* to each STA, encrypted using the STA's *KEK*, with a MIC using the STA's *KCK*.
2. The return message activates the new GTK.

3.3 Robust Security Networks

Most of the material in this section was also presented by the author in "*Securely Deploying IEEE 802.11 WLANs*" [24], at the *AusCERT Asia Pacific Information Technology Security Conference (AusCERT2007)*, in May 2007.

These advances in the data security of wireless LAN networks now make commercial and government use viable. Only the RSN is capable of providing a DSD[3] Approved Cryptographic Algorithm (DACA) [128]. RC4 is not a DACA, so WEP-40, WEP-104 and TKIP are not appropriate. Only the *IEEE Std 802.11i* Robust Security Network Association (RSNA) provides negotiation of encryption algorithms, thus allowing for new algorithms in the future.

However, merely conforming with the new amendments is not sufficient to successfully implement an RSN and many mistakenly believe that simply turning on WPA or WPA2 is sufficient to meet the requirements of an RSN, necessary in government and many commercial situations.

The mere existence of WPA2 certification of equipment is not sufficient to provide a RSN, since compliance with IEEE Std 802.11i does not mandate a RSN in operation. There is a critical difference between *RSNA-capable* devices and *RSNA-enabled* devices.

WPA2 certified equipment *can* implement a RSN, but not necessarily so. WPA2 certification requires backward compatibility with WPA. Despite TKIP being an optional protocol for RSNs, WPA/TKIP is *not* a RSN. In addition to this, any Wi-Fi certification demands backward compatibility with WEP — but WEP is incompatible with a RSN and some vendors do not keep the two separate.

For example, the Temporal Key Integrity Protocol (TKIP) is one of the two defined RSNA data confidentiality and integrity protocols in the amendment to the standard. As such, its use, in itself, does not preclude a RSN. However, in practice, selecting TKIP in WPA2 certified equipment often also invokes WPA compatibility to allow connections with WPA certified equipment, which can only use TKIP or WEP and do not implement all of the mandatory elements of IEEE 802.11i — and so this is not a RSN [24].

[3]Australian Government, Department of Defence, Defence Signals Directorate.

3.3.1 RSN and RSNA

IEEE 802.11i introduces the ***robust security network*** **(RSN)** — "A security network that allows only the creation of ***robust security network associations*** **(RSNAs)**" [12, Definitions 3.106], being associations "if the procedure to establish authentication or association between them includes the 4-Way Handshake" [12, Definitions 3.107] — the 4-way authentication exchange defined in this amendment to the standard. These are two critical definitions.

3.3.2 RSNA Requirements

IEEE 802.11i is not a standard in itself. It is an amendment to the IEEE 802.11 standard. As such, it does not have its own Protocol Implementation Conformance Statement (PICS), although it does amend the IEEE 802.11 PICS. These amendments make RSNA **optional**, but, if implemented, then implementation of the RSN Information Element (IE) and CCMP is mandatory.

That is not to say that CCMP must be used, but it must be implemented "in all IEEE 802.11 devices claiming RSNA compliance" [12, section 8.3.1]. It is important to note that a RSN does not, by definition, specify any particular encryption algorithm, but depends solely on the associations it allows.

A RSN ***only allows*** RSNAs. RSNAs can use either TKIP or CCMP, along with RSNA establishment, termination and key management procedures. To draw the distinction clearly, a network that only has RSNAs is not necessarily a RSN, because it may still allow pre-robust security network associations (pre-RSNAs).

The other mandatory element is the RSN IE. It includes the list of available pairwise (unicast) ciphers and the single groupwise (multicast/broadcast) cipher. The pairwise cipher is used for communication only between the two parties of a single association, while the groupwise cipher is used for multicast or broadcast communications with many or all associated parties. That is, a unique pairwise cipher and key is used between the AP and each STA, while the groupwise cipher and key is used for broadcasts from the AP to all STA. An AP in a RSN must include this RSN IE in its Beacon and Probe Response frames and any STA wanting to participate in a RSNA must include the RSN IE in its Association Request frames [24].

3.3.3 RSN and TSN

Thus, a RSN only allows RSNAs; and any STA in a RSNA must implement CCMP, may optionally implement TKIP, and can use either (if implemented) TKIP or CCMP. But for a network to be a RSN it must not allow Pre-RSNAs. So an AP in a RSN must not associate with pre-RSNA STAs — STAs that do not have the RSN IE in the Association Request — and it must include the RSN IE in its Beacon frames, showing the group cipher suite as CCMP or TKIP, but **not** WEP [12, p. 5].

A network that allows the creation of pre-RSNAs is not a RSN. A network that allows the creation of pre-RSNAs as well as RSNAs is called a ***transition security network*** **(TSN)** and can be identified by the RSN IE of Beacon frames showing that the group cipher suite is WEP [12, p. 6]. Figure 3.1 compares the functionality of RSNs and TSNs within the WLAN environment. In a TSN, a RSN STA shall include the RSN IE in its Association Requests. In an infrastructure network, as all associations are with the AP, both a TSN and RSN require a RSNA-capable AP. A RSNA-capable AP configured to operate in a TSN must include the RSN IE in its Beacons and can associate with both RSNA and pre-RSNA STAs. Pre-RSNA STAs will ignore the unknown RSN IE from the AP and will not have a RSN IE in their requests [24].

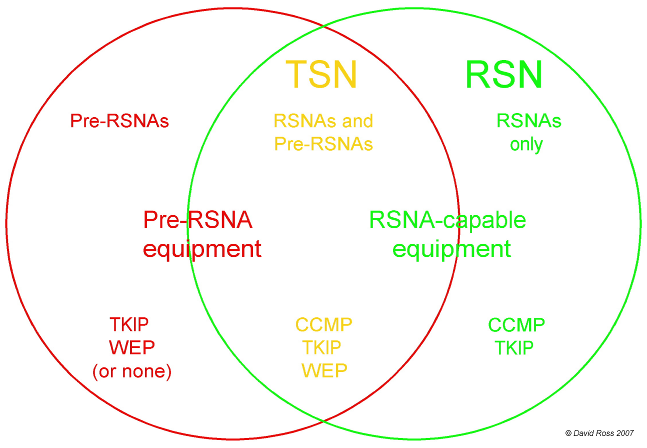


Figure 3.1: RSNs and TSNs within the WLAN Environment [24]

When an AP in a TSN receives an Association Request without an RSN IE, its IEEE 802.1X Controlled Port shall initially be blocked. The IEEE 802.1X Station Management Entity unblocks the IEEE 802.1X Controlled Port when

WEP has been enabled [12, p. 67]. In this case, a STA using WEP can have the same network access as a STA using a RSNA, thus bringing the security for the entire network down to the level of simple WEP protection. Mixed modes involving WEP offer very poor protection. In these cases, additional measures, such as separate 'via-WEP' subnets, separate WEP APs (or separate SSIDs within an AP [129]) or other network access control should be employed.

The possible cipher suite combinations in the RSN IE are shown here [12]:

Pairwise	Groupwise	
CCMP	CCMP	(RSN)
CCMP	TKIP	(RSN)
CCMP, TKIP	TKIP	(RSN)
TKIP	TKIP	(RSN)
CCMP	WEP-40 or WEP-104	(TSN)
CCMP, TKIP	WEP-40 or WEP-104	(TSN)
TKIP	WEP-40 or WEP-104	(TSN)
UseGroup	WEP-40 or WEP-104	(TSN)

The combination of TKIP as the pairwise cipher and CCMP as the group cipher is not permitted. An AP can "Use group cipher suite" for the pairwise cipher suite if it does not support any pairwise cipher suites [12].

3.3.4 Wi-Fi Protected Access (WPA)

With the problems of WEP apparent to all, the Wi-Fi Alliance could not wait for IEEE 802.11i to be ratified. It released Wi-Fi Protected Access (WPA), based on a snapshot of the IEEE 802.11i/D3 draft, as a security solution that fixed the WEP vulnerabilities in the original IEEE Std 802.11 for WLANs operating in infrastructure mode. WPA is not certified for ad hoc mode. WPA uses the Temporal Key Integrity Protocol (TKIP) from IEEE 802.11i for encryption, including the message identity check (MIC) and per-packet keying, but the WPA Information Element (IE) is not the same as the IEEE 802.11i IE [24].

Having evolved from a TGi draft, WPA raises ambiguities in comparison to the final IEEE 802.11i release. In TKIP-Only Mode, it uses TKIP for both the pairwise and groupwise suites and so IEEE 802.11i's definition for a RSN — "A RSN can be identified by the indication in the RSN IE of Beacon frames that the group cipher suite specified is not wired equivalent privacy (WEP)." [12, Definitions 3.106] — implies that WPA supports a RSN. However, it is clear from the minutes of the meetings of TGi that this is not the case. WPA only implements a subset of a RSN. It does not implement the mandatory CCMP [12, Annex A PICS PC34.1.2.1] (or Ad Hoc networks or support for Quality of Service (QoS) or support for pre-authentication), thus is pre-RNSA. It forms, by definition, a TSN.

Wi-Fi certified WPA devices are not *intended* to service a mixture of WEP and WPA. The Wi-Fi WPA certification demands that the *default* configuration not support this. However some vendors offer this sort of functionality 'on top of' WPA certification, e.g. Cisco Aironet APs allow a "WPA Migration Mode" where both WPA and WEP STAs concurrently associate to the same AP. This mode of operation clearly forms a TSN. SMC offer this functionality on their SMC2804WBR 'Barricade **g**' wireless router [24].

Devices certified for WPA Personal Mode offer only pre-shared key (PSK) authentication, requiring manual configuration of a pre-shared key. No authentication server is used. Whereas, devices certified for WPA Enterprise Mode offer both PSK and IEEE 802.1X/EAP authentication [23].

3.3.5 WPA2

WPA2 is the Wi-Fi Alliance certification of an implementation of IEEE 802.11i. WPA2 has all of the mandatory requirements of IEEE 802.11i "but differs slightly to allow for interoperability" [127] with WPA. All Wi-Fi certified WPA2 devices are interoperable with Wi-Fi certified WPA devices. WPA2 provides both TKIP (using RC4) and CCMP (using AES) and is available in both infrastructure mode and ad hoc mode [24].

3.3.6 When 'WPA2' implements a TSN

The difference between IEEE 802.11i and WPA2 is small but non-trivial. Although IEEE 802.11i states a RSN can optionally implement and use TKIP as well as the mandatory CCMP, when a WPA2 device is configured to allow TKIP as well as CCMP, it often implements TKIP in the same manner as the WPA certification and runs a mixed IEEE 802.11i and WPA network — a TSN. The implementation of this mixed WPA/WPA2 mode is mandatory for Wi-Fi WPA2

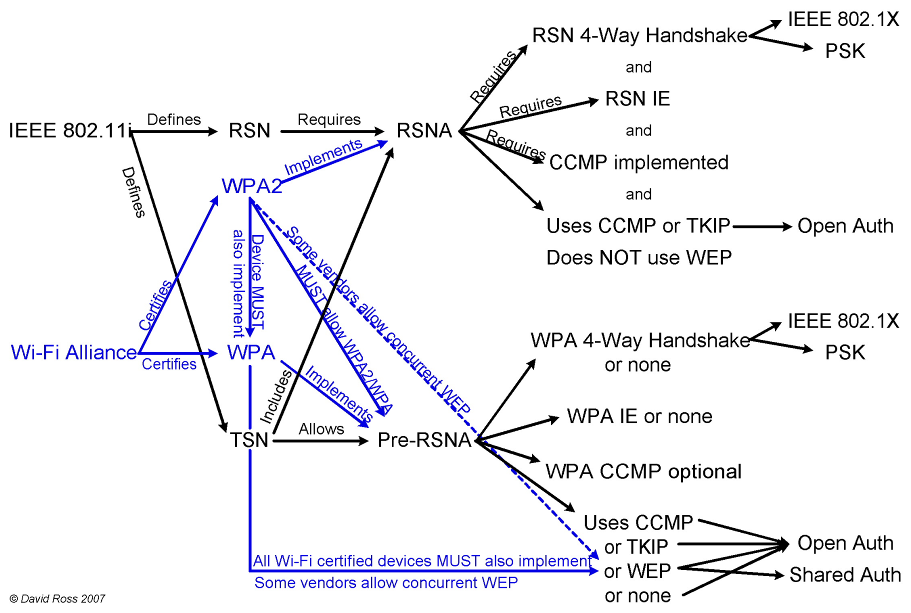


Figure 3.2: Differing requirements for RSN, TSN, WPA and WPA2 [24]

certification and so is available on *all* WPA2 certified devices. This is becoming less of an issue, as vendors increasingly offer separate configuration options to allow TKIP for WPA2, separate to support for 'WPA-compatibility' — at least on their commercial-grade equipment. Consumer-grade equipment often still combines the two as a single option.

The Wi-Fi Alliance's WPA2 certification, like WPA, checks that, ***in the default configuration***, devices in WPA2 mode do not support WEP devices at the same time. However, every Wi-Fi certified device must support WEP, as it is a base interoperability requirement for Wi-Fi certification. Many vendors offer modes that permit a mixture of WPA2 and WPA or WPA and WEP or even all three at once, then these devices can still be WPA2 certified but in these other modes will form a TSN. Figure 3.2 illustrates the differences between RSN, TSN, WPA and WPA2.

Thus, the only WPA2 mode that supports a RSN is the so-called "WPA2-Only Mode". Any other mode that permits both RSNAs and pre-RSNAs will form a TSN. The Wi-Fi WPA2 certification requires "WPA/WPA2 Mixed Mode" to be implemented. In WPA/WPA2 Mixed Mode, the AP provides both CCMP and TKIP for the pairwise ciphers and uses TKIP as the group cipher suite to allow both WPA and WPA2 STAs to associate on the same SSID [24].

Joseph Davies on Microsoft's TechNet offers "WPA2-certified wireless equipment is also compatible with WPA and WEP. You can have a mixture [of] WPA2, WPA, and WEP wireless devices operating in the same environment [130]". Such a configuration is clearly a TSN.

Netgear Wi-Fi certified APs can also support both WEP and WPA/WPA2 clients at the same time, to support the transition of WEP-based wireless networks to WPA/WPA2. Again, this is a TSN.

Any configuration involving RSNAs (i.e. WPA2) and ***permitting*** pre-RSNAs (WPA or WEP) is a TSN — irrespective of whether or not and pre-RSNA equipment is actually present or associated with the network. If pre-RSNAs are permitted then it is a TSN. This is important because it does not matter if all the devices are using AES-CCMP, *if the AP will allow a pre-RSNA association, then the security may be compromised.*

The significant risk here is where an organisation has been using a mixed-mode WLAN, and converting all devices to use AES-CCMP, erroneously believes the WLAN is a RSN, when there are still APs that would accept a pre-RSNA. If an infrequently-used device were to be powered up and associate with WEP, within minutes an attacker could determine the WEP key and gain access to the network [24].

3.3.7 Multiple 'WPA2' Certifications

Another difficulty has emerged with the Wi-Fi Alliance changing its WPA2 certification. Part of the security of a RSN relies on the (unspecified) Extensible Authentication Protocol (EAP) providing **mutual** authentication. Some EAPs do not provide this. For example, EAP-MD5 does not, while EAP-TLS does [131]. The original Wi-Fi Alliance WPA2 certification required EAP-TLS to be implemented. The latter certification has expanded the list of protocols, but EAP-TLS is no longer mandatory for all types of devices.

The problem is that the certification is the same, *WPA2*, and users cannot readily identify whether products were certified under the first method or the second without specifically asking the vendor for a copy of the certification certificate.

3.4 Attacking WPA2

With the lack of empirical data available as to the actual possibility of vulnerabilities in common-off-the-shelf devices, such as might be used in SOHO situations (consumer-grade devices), allowing simultaneous WEP and WPA2 associations when configured as "WPA2", the author conducted a series of tests on a number of combinations of vendors' devices with various chipsets.

These tests are detailed in Appendix B and were also presented by the author in "*Securely Deploying IEEE 802.11 WLANs*" [24], at the *AusCERT Asia Pacific Information Technology Security Conference (AusCERT2007)*, in May 2007.

The initial tests were performed with **Microsoft** Windows clients and a **Linksys** WRT54G AP. The WRT54G allowed the selection of WPA2 with AES or TKIP+AES or WPA with AES or TKIP or WEP, with all possible combinations of authentication and key-modes. The Windows clients offered a simplified range of options with WPA2, WPA2-Personal, WPA, WPA-Personal and WEP.

All legal combinations of authentication and encryption were tested and the clients were able to connect for all matching combinations and, as required and expected, mixed encryption modes could not connect. None of the variations of clients, for either **Atheros** or **Texas Instruments** chipsets, combined with either **Microsoft** or **Linux** driver software, on any of the platforms, could successfully associate with WEP with any of the APs correctly configured for WPA2 (only). These typically were indicated by the client STA finding no suitable AP (with intersecting sets of protocols) in the responses to its probes. When Linux clients were used, so that association could be forced by the client STA, the AP would typically respond with the association denied for an unknown reason [24].

3.4.1 Weak Configurations

Of all the test APs, as given in Appendix B, only the **SMC** SMC2804WBR "Barricade g Wireless Broadband Router" offered a WEP/WPA combined mode. The SMC2804WBR was configured for WEP/WPA combined mode and simultaneous associations were successfully made with both WPA and WEP clients. Both the WPA and WEP clients were able to access the network behind the AP.

It was a trivial matter to capture the traffic needed to use with **aircrack** to recover the WEP key with a separate "intruder" device. It was found that the WEP key allowed an attacker to get to the entire wired LAN behind the

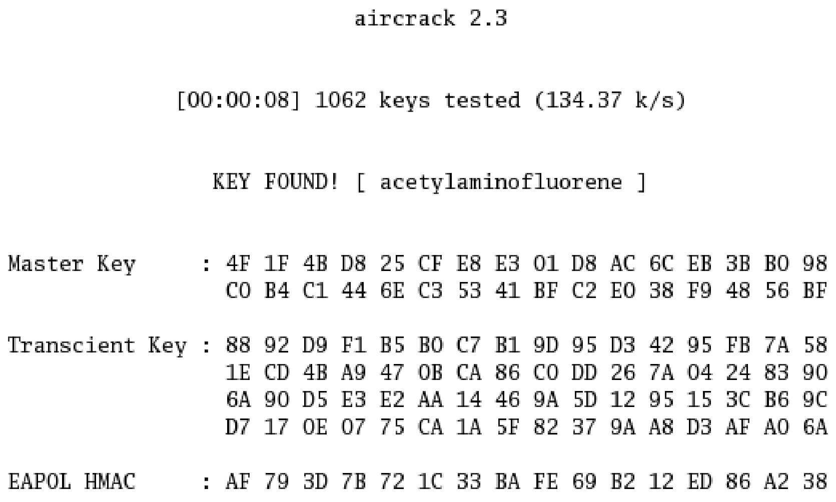
SMC2804WBR, severely compromising the wired LAN and by setting the AP as the default gateway, all other subnets and the Internet service.

Curiously, although the WEP key allowed an attacker to get to the entire wired LAN, we were not able to access the WPA devices using the same AP for the same wired LAN. Again, this is not a clandestine attack — the SMC2804WBR has to be configured as WEP/WPA to allow this attack. This is a clearly dangerous configuration [24].

3.4.2 Using Pre-Shared Keys with WPA and WPA2

Note that in the simplest case, for small private networks, there is the option of using a Pre-Shared Key (PSK) with WPA2 (WPA2-PSK). The Wi-Fi Alliance calls this "WPA2-Personal".

The remaining issue, was the use of 'Pre-Shared Key' (PSK) mode in all of these tests. A common comment in Internet forums is that the pass-phrase should be at least 20 characters long. However this should be random characters, or there will be insufficient entropy. Tools like ***aircrack*** can break non-random words with a dictionary attack — in seconds. Figure 3.3 shows how it took **aircrack** only eight seconds to find our reasonably long (19 characters and 20 characters), but very poorly chosen pass-phrases, "acetylaminofluorene" and "acetylcholinesterase" that were used for WPA and WPA2 key pass-phrases throughout these tests.



```
                              aircrack 2.3


                 [00:00:08] 1062 keys tested (134.37 k/s)


                    KEY FOUND! [ acetylaminofluorene ]


Master Key     : 4F 1F 4B D8 25 CF E8 E3 01 D8 AC 6C EB 3B B0 98
                 C0 B4 C1 44 6E C3 53 41 BF C2 E0 38 F9 48 56 BF

Transcient Key : 88 92 D9 F1 B5 B0 C7 B1 9D 95 D3 42 95 FB 7A 58
                 1E CD 4B A9 47 0B CA 86 C0 DD 26 7A 04 24 83 90
                 6A 90 D5 E3 E2 AA 14 46 9A 5D 12 95 15 3C B6 9C
                 D7 17 0E 07 75 CA 1A 5F 82 37 9A A8 D3 AF A0 6A

EAPOL HMAC     : AF 79 3D 7B 72 1C 33 BA FE 69 B2 12 ED 86 A2 38
```

Figure 3.3: Pre-Shared Key (PSK) broken in 8 seconds [24]

There is a critical difference between *RSNA-capable* devices and *RSNA-enabled* devices. It has been shown that conforming with the IEEE 802.11i amendment does not necessarily enforce a RSN and the WPA2 certification of equipment is not sufficient to provide a RSN, since compliance with IEEE 802.11i does not mandate a RSN in operation.

WPA2 certified equipment will implement a RSN, if properly configured to do so, but this may not be the default. With WPA2 certification requiring backward compatibility with WPA, it is likely that WPA2 devices will default to a TSN. In addition to this, many vendors provide various other mixed pre-RSNA modes of operation [24].

Even if using a RSN, control and management frames remain unsecured, as are the link layer headers, still providing multiple attack vectors into these networks.

Chapter 4

Remaining Unprotected Protocols

This chapter differentiates the current state of the art, given in the previous chapter, and the remaining unprotected components of the WLAN protocols, providing vulnerabilities that may be exploited by adversaries to corrupt normal operations. It provides more detailed operational information and provides a gap analysis between the realities of IEEE 802.11 networks and the potential for malicious activities. This describes the need for and various applications of this research, described in the following chapters.

Analysis of IEEE 802.11 security is frequently based on current or known issues, such as those already described in the current literature considered in Chapter 2. While such issues are probably the most pressing for time and certainly at the forefront of the press, a thorough analysis of the security issues for any technology requires a thorough understanding of the operation of the technology in both normal and abnormal states.

Thus, this chapter begins from the physical medium, its characteristics and intrinsic issues, and how the IEEE 802.11 protocols deal with those issues, as well as the implications of operating the protocols in this way. In this manner, we build up a picture of the remaining security issues either not currently handled by the protocols to date or even as a direct result of the operation of the protocols as they currently stand.

4.1 Radio Communications

Transmitting IEEE 802.11 STA emit RF energy. While various antenna designs can concentrate this energy onto a particular path, some of the energy is radiated in all directions and unless captured in purpose-built shielded facilities, is capable of being detected by any party, friend or foe, in any direction.

Receiving STA absorb RF energy. While, in theory, it is possible to determine if a STA is or is not, or stops or starts, receiving a signal based on the remaining reflected and ambient energy, in practice this requires a highly controlled environment and practical IEEE 802.11 deployments cannot determine if any party, intended or not, is receiving a signal, unless the party responds to the signal.

These two factors combine to make it very easy for an adversary to covertly listen to radio communications, while making it difficult to assure any degree of success in radio communications with intended parties.

4.1.1 Interference

Even if two STA are successfully communicating over the WM, individual signals, or parts of signals, may be damaged, altered or destroyed by changes in the environment, movement of the STA, movement of objects in the transmission path, multi-path distortion, interfering signals from other sources or other technologies, or even by active jamming of the frequencies by an adversary.

4.1.2 Collisions

Just as for wired networks, if two or more STA transmit at the same time, with sufficient strength, the resulting collision will render the signal unreadable. However, unlike wired networks, the sending wireless STA typically cannot listen at the same time as transmitting and so have no way to detect if a collision has occurred. Also, a collision may only occur locally at the receiving STA, where another transmitting STA, out-of-range of the first transmitter and hence hidden from that transmitter, also transmits within range of the receiving STA.

As such, the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) methods of IEEE 802.3 wired networks are unsuitable for IEEE 802.11 wireless networks, where Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) methods must be employed. This process uses physical and virtual carrier sensing and a system of deferred access and truncated binary exponential backoffs, along with the use of positive acknowledgements for all unicast traffic.

4.1.3 Acknowledgements

To support this, for all directed (unicast) traffic, IEEE 802.11 uses 'immediate' positive acknowledgements (ACK) [5]. These ACKs are Control frames sent by the recipient of a directed frame, back to the original sender, whenever a directed frame is *successfully* received. If no ACK is returned, a collision or damaged/lost frame is assumed and the frame is retransmitted. As such, every legitimate STA receiving a unicast frame will announce this fact to the world by transmitting an ACK back to the sender. In this manner, the transfer of every MSDU actually requires two successful transmissions, both the PPDU and the return ACK — increasing the opportunity for failure and subsequent retransmission.

4.1.4 Deferred Access

When a STA wants to send a message, it must defer access until it senses the medium is idle for the period of an interframe space. The actual lengths of the interframe spaces depend on the PHY layer, but are in the order (from shortest to longest):

- SIFS — Short Interframe Space, before a Control frame for a related preceding frame, such as an ACK or CTS, has priority over everything;
- PIFS — PCF Interframe Space, under the Point Coordination Function, has priority over DCF traffic;
- DIFS — DCF Interframe Space, under the Distributed Coordination Function; and
- EIFS — Extended Interframe Space, after a frame error, gives problem nodes the lowest priority.

4.1.5 Slots

When one STA sends a message to another STA, this signal will take a finite amount of time to propagate to the intended destination and to every other STA in this broadcast medium. If a third STA were to physically sense the medium as idle, and where to immediately attempt to transmit a frame, it may well occur that there is already a frame destined to this or some other STA already being propagated through the medium, resulting in a collision destroying both frames. To minimise the chances of such collisions, all STAs are required to transmit only at the beginning of a slot boundary. The slot times a different for the various physical implementations, but are designed to ensure that if any other STA has begun transmission on the previous slot boundary, then that signal will have propagated to all other STAs before the next slot boundary.

4.1.6 Backoff

When a STA wishing to transmit, senses the medium, and finds it busy, it must wait until the medium becomes idle for the appropriate interframe space. However, if all STAs in the network wishing to transmit a frame, also waited that same amount of time, then all of these STAs would begin to transmit at roughly the same time, causing a high collision load in the network. For this reason, any STA wishing to transmit after it has sensed the medium to be busy, must wait the appropriate interframe space plus a random number — chosen from the current Contention Window (CW) — of backoff slots to avoid simultaneous transmission by all waiting STAs.

4.1.7 Physical and Virtual Carrier Sense

The carrier sense systems of wireless networks include both physical and virtual carrier sense mechanisms. The physical carrier sense mechanism — the Clear Channel Assessment (CCA) function — performs the same function as the physical carrier sense in a wired network. However, in a wireless system, not all of the signals necessarily reach all of the STAs. Just physically sensing the medium is insufficient to avoid broadcasting a signal into an already-busy area of the WM. So IEEE 802.11 wireless systems also employ a virtual carrier sense mechanism — the Network Allocation Vector (NAV). If the STA is transmitting (and therefore the WM around the STA is busy) *or* the STA's physical carrier sense shows the WM to be busy *or* the NAV is non-zero (the STA 'knows' there must be a transmission occurring somewhere), then the WM is 'busy'.

4.1.8 Network Allocation Vector (NAV)

The NAV is updated whenever a STA receives a packet *not* addressed to it. The NAV is updated with the duration field from the received packet if this is greater than the value already in the NAV. Broadcast packets have a duration of zero (nothing follows). Directed (unicast) traffic has the value of a SIFS plus the time to transmit the return ACK (the WM is assumed busy for a SIFS + ACK after this frame). RTS and CTS frames, described below, update the NAV with the time to reserve the WM.

4.1.9 RTS/CTS Control Frames

If three nodes are arranged in such a way that the two outer nodes can both communicate with the central node, but cannot detect transmissions from each other, then if one of the outer nodes is transmitting to the central node, both this node and the central node can physically sense that the medium is busy. However, the other node, which is out of range of the transmitting node, cannot physically detect a busy medium. If this third node, were to also attempt to transmit to the central node, even though there is no activity in the WM around this node, this will cause a collision with the active signal already being received at the central node. The outer nodes each act as a 'hidden node' with respect to the other.

IEEE 802.11 networks can use CTS/RTS control frames to reserve time on the medium for a transmission. The STA wishing to send a frame will first issue a Request To Send (RTS) to reserve the medium for the amount of time required to transmit the frame. Once again, a hidden node that cannot detect the transmitting STA, would also not detect that STA's RTS. Therefore, the STA receiving the RTS issues a Clear To Send (CTS) frame, confirming the reservation of the medium for the remaining amount of time left from the RTS.

Thus, any STA within range of the sender will see the RTS and will know the medium will be busy for the period of time in the request — and any STA within range of the receiver will see the CTS and, even if it did not see the original RTS, will know that the medium will be busy for the period of time in the CTS, which will be a SIFS and the time to transmit the CTS *less* than that given in the RTS.

This mechanism, while resolving the hidden mode problem and improving network response in those situations, provides a vector for a DoS attack. When an attacker can spoof CTS frames, it can cause other STAs in the network to falsely reserve time for a busy medium. In addition, where an attacker can spoof RTS frames sent to the AP and the AP responds with a CTS, as the AP is in communication with *all* STA on the network, then this attack will cause every STA in the network to falsely reserve time for an allegedly busy medium.

4.2 Establishing Communications

Irrespective of the physical layer protocols in use, some of the specifics of which are discussed later, or the mode of operation intended, there are a number of processes any STA must undertake before it can attempt to establish wireless communications. These processes typically involve the exchange of various Management frames, often completely unprotected, to transfer the necessary configuration information to complete the task.

4.2.1 Message Types

There are three principal types of IEEE 802.11 messages. These are Data messages, Control messages and Management messages. Data messages provide the service for the upper layer protocols, carrying the MSDUs to their destination MAC components. Control messages, as discussed in the previous section [4.1], are used to control the actual process of transferring messages at the MAC level, such as the positive acknowledgements and the RTS/CTS reservations discussed above.

The Management messages are used to set up and tear down the communications for a STA. The basic Management frames include Beacon frames, Probe Request and Probe Response frames, Authentication and Deauthentication frames, Association Request and Association Response frames, along with Reassociation Request and Reassociation Response frames, as well as Disassociation frames and Action Management frames. State 1 Management messages are used to access the basic station services (SS), State 2 Management messages are used to access distribution system services (DSS) and State 3 Management messages add services such as QoS, station-to-station link services (e.g. Direct Link Service) and block acknowledgements.

4.2.2 Frame Classes

There are three classes of IEEE 802.11 frames. These should not be confused with the three message types or their various subtypes and frame formats.

The three frame classes a related to the current communications state of the STA. These states describe the relationship currently existing between the source and the destination STA and are one of:

1. **State 1**: unauthenticated and unassociated,
2. **State 2**: authenticated and unassociated, or
3. **State 3**: authenticated and associated.

Communications are initiated in **State 1**. A STA currently in **State 1** with its intended destination may only send **Class 1** frames. Class 1 frames are permitted in any state, but are the only frames permitted when a STA is in State 1 with respect to its intended destination. Class 1 frames include a subset of all three of the principle types of messages. This includes the basic control frames; the management frames related to beacons, probes, authentication and deauthentication; and a limited subset of the data frames.

When a STA in **State 1** with respect to its intended destination authenticates to the destination, the relationship between the STAs moves to **State 2**, authenticated but not associated. In **State 2** both **Class 1** and **Class 2** frames are permitted. Now that the STA is authenticated, the Class 2 frames provide additional management capabilities with all of the Association, Reassociation and Disassociation management frames now available. These additional frames manage the associations between a STA and APs in infrastructure networks.

When a STA in **State 2** successfully Associates with an infrastructure network AP, the communications move to **State 3** allowing all Class 1, 2 and 3 frames. Class three frames include all of the remaining subtypes of Control, Management and Data frames; and represent the entirety of the IEEE 802.11 services .

4.2.3 Network Discovery

Before a STA can commence communications over an IEEE 802.11 network, it must first discover what networks are available. It can do this using either a passive scan, listening for Beacon frames from any active networks present, or an active scan, by sending Probe Request frames and waiting for any Probe Response frames in return [129]. This is typically controlled by the firmware and should also be governed by the regulatory domain applicable at the location where the STA is operating, as a number of regulatory domains do not permit active scanning on some or all of the IEEE 802.11 frequencies.

A passive scan relies on the information being broadcast in Beacon frames from any active networks within range. These Beacon frames contain most of the information a STA needs to join a network or move about from one BSS to another and are broadcast many times per second. The format of these Beacon frames is given in Figure 4.1.

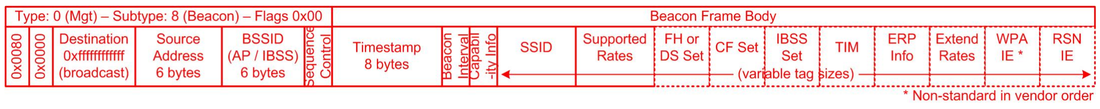


Figure 4.1: IEEE 802.11 Beacon Frame Format

* Note that in each of these diagrams where a non-standard information element (IE), such as the WPA IE shown in Figure 4.1, is included by a vendor, these should occur *after* the standard information elements. However while all vendors appear to place the standardised information elements in the correct order, non-standard information elements, such as the WPA IE appear in various vendor-dependent orders. While the WPA IE depicted here, typically appears in the place of the RSN IE, which would correctly place the vendor-specific IE at the end, the author has also seen instances of frames containing both the WPA IE and the RSN IE in the order shown in Figure 4.1.

As Figure 4.1 shows, these regular sub-second Beacon frames provide the majority of information a STA, or adversary, needs to be able to join the network. This includes the network capabilities, SSID, transmission rates and other settings.

The SSID is the network name of up to 32 octets. In an infrastructure network the SSID is broadcast in the beacon frames by the access point. In an ad hoc network, an IBSS, the random SSID chosen is broadcast by all the members of the independent service set.

In a BSS or IBSS this network name is different to the Basic Service Set

Identification[1] (BSSID), which is the IEEE 802 48-bit MAC address of the STA contained in the AP or the IBSS logical (random pseudo-) IEEE 802 48-bit MAC address. In an ESS, this network name is the extended service set identifier and although referred to as "ESSID" once [5, p. 25] and "ESS ID" once [5, p. 25], is simply the SSID of the ESS or IBSS, but is often referred to as the ESSID or NetID, although the latter is not a standardised term.

A zero-length SSID is the broadcast SSID. Whereas, a value of all 1's is used to indicate the broadcast BSSID [5]. A null SSID, consisting of one or more ASCII NULL characters (0x00), as opposed to a zero-length SSID, is referred to as a "hidden" SSID for what is sometimes referred to as a "closed" network.

In an infrastructure network access points can be configured not to broadcast the SSID in the beacon frames by setting it to null in these beacon frames. As discussed in Chapter 2, this is a common but ineffective security measure employed for many networks.

Because the SSID is often displayed and entered by the users of the devices, it typically consists of printable ASCII characters. The only requirement in the standard is that it be up to 32 binary octets and therefore can take on any value. However in large and diverse networks with a heterogeneous mixture of vendor equipment, the various limitations from the different vendors for methods to enter the SSID will usually dictate an SSID consisting of only printable ASCII characters.

While a passive scan is innocuous enough, an active scan will broadcast a series of Probe Request frames, often including probes for all preferred networks with any "hidden" SSIDs broadcast in the clear, as well as the "Broadcast SSID" (zero-length SSID) to get any responses available. The format of these Probe Request frames is given in Figure 4.2.

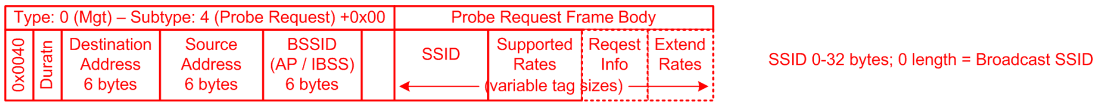

Type: 0 (Mgt) – Subtype: 4 (Probe Request) +0x00
Probe Request Frame Body
0x0040
Duratn
Destination Address 6 bytes
Source Address 6 bytes
BSSID (AP / IBSS) 6 bytes
SSID
Supported Rates
Reqest Info
Extend Rates
(variable tag sizes)
SSID 0-32 bytes; 0 length = Broadcast SSID

Figure 4.2: IEEE 802.11 Probe Request Frame Format

The results are returned in a Probe Response frame The format of these

[1]The IEEE Std 802.11-1997 [5] defines "identification" for BSSID, but uses "identifier" in the text, e.g. p. 38. However, IEEE Std 802.11-2007 [122] uses only "identification" for BSSID and only uses "identifier" for SSID.

Probe Request frames is given in Figure 4.2 and the Probe Response is given in Figure 4.3.

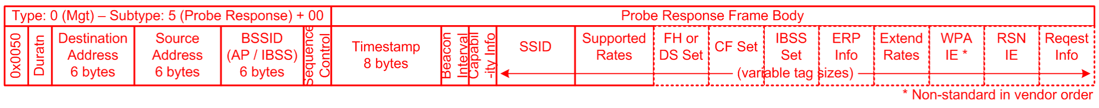


Figure 4.3: IEEE 802.11 Probe Response Frame Format

Note that the original IEEE standard does not include the Traffic Indication Map (TIM) information element in probe response frames, however a number of actual implementations do include the TIM elements in their probe response frames.

4.2.4 Authentication and Association

Authentication and association in wireless networks are critical components protecting the confidentiality and integrity of the data traversing not only the WLAN itself, but also the hardware supporting it, either the wired networks behind an infrastructure WLAN or the individual devices participating in an ad hoc WLAN.

The use of unsecured WLANs for criminal activity, from misdemeanours to crimes, or even possible acts of terrorism [132], is ever on the increase. Neighbourhood or passing thieves steal bandwidth. Other criminals steal company or personal data, or even identities. Yet others use unsecured WLANs to keep their own identity and location hidden. All this nefarious activity is escalating exponentially as more and more businesses and residences install WLANs with ever-increasing range and bandwidth.

Many projects are under way to make wireless networks even more robust. The US DoD Wireless Adaptive Network Development project (WAND) [133], is part of the overall project Wireless Network after Next (WNaN) program [134], to provide cheap USD$500, spectrum-agile, disruption-tolerable communications networks for up to 1000 nodes, able to search-out and jump to unused spectrum to avoid interference or jamming. WNaN is also planned to allow for simultaneous 100baseT access.

By the time these advances make it to consumer electronics, the potential for criminal or terrorist activity utilising the epidemic of unsecured WLANs may well have become a law enforcement and national security nightmare.

4.3 Authentication

For most practical purposes, an entity wishing to access resources should be identified, authenticated and authorised. In *identification*, a *claimant* asserts an *identity*, then in *authentication*, a *verifier* verifies the claimed identity asserted by the claimant, by *validating* the *credentials* presented for that previously *registered* identity [135]. Sometimes, the credentials are provided by a *trusted third party*. Finally, access is granted to only those resources for which the identity has already been *authorised*.

Authentication must not be confused with authorisation. *Authorisation* is the *granting* of *access privileges* to an identity for the purposes of *access control* to resources [135].

The claimant, or some other *applicant* on behalf of the claimant — a *subscriber* in the terminology of a Public Key Infrastructure (PKI) — must first undertake an *enrolment* process, with a *registration authority* that verifies the identity to some standard (including the standard of no verification at all) and registers the identity and the credentials associated with that identity. The registration authority stores these details, typically in some form of *directory*. *Validation engines* reference such a directory when a verifier needs to validate the enrolled identity.

4.3.1 Authentication Credentials

A credential is evidence used to verify an identity. Identity can refer to a physical entity or a logical entity. Credentials take many forms but are grouped onto three main types:

- Something you know,
- Something you have, or
- Something you are.

Using one or more of these types of credentials provides different authentication *factors*. Using different authentication factors increases the *Identity Authentication Assurance Level* (IAAL) that the assertion is valid. This combined with the *Identity Registration Assurance Level* (IRAL) determined from the strength of the original registration identity validation process, gives the overall *Authentication Assurance Level* (AAL), the level of confidence that identity is that of the claimant [136].

Passwords

A password, ‘pass-phrase’, Personal Identification Number (PIN) or other memorised credential is an example of ‘something you know’. It is not a very strong credential as knowledge, once revealed, is easily duplicated and many entities may be given the same ‘shared secret’.

Tokens

A physical key, access pass (identity card, passport or other documents), cryptographic token, smart-card and the like represent ‘something you have’. While much harder to duplicate than a secret, these tokens are also more visible and more easily stolen than a secret.

Biometrics

A biometric is a sample of ‘something you are’ — face, fingerprint, retina pattern, Deoxyribonucleic Acid (DNA) profile. Biometrics are unique in that the credential includes intrinsic Identity Registration Assurance, however as some biometrics are relatively easy to covertly steal compared to a token and as some biometric scanners are relatively easy to subvert or bypass entirely, unsupervised biometric authentication only provides low Identity Authentication Assurance and is not suitable in many situations.

Digital Certificates

A digital certificate is the binding of a digital encryption key to a digital identity and attests that the named identity, the certificate ‘subject’, is the holder of the matching asymmetric private key to the asymmetric public key contained in the certificate. Subject to the relative strengths of the asymmetric encryption being used, the certificate effectively transitively asserts that “only the subject can decrypt anything that you encrypt with this asymmetric public key” and “anything that this asymmetric public key successfully decrypts can only have been encrypted by the subject”.

The certificate is created by a mutually trusted third party called a Certification Authority (CA). The CA performs the identification, enrolment and registration functions of a Registration Authority (RA), described above, either itself or delegated to one or more separate subordinate dedicated RAs.

The integrity of the contents of the certificate is protected using a cryptographic hash algorithm, which is then digitally signed using an encryption algorithm with the CA's private key. Thus, the subject's public key is now asserted by a certificate that can be validated by decrypting the signature with the CA's public key — which if not known can be obtained from the CA's own CA-certificate, possibly issued by some higher CA, and so forth, until the root CA — the root of trust — is reached with a self-signed (subject and issuer are the same) root CA-certificate. The success of any hierarchical PKI relies on the trust in the root. A validation engine must explicitly trust one of the CA-certificates in the chain from the subscriber's certificate in order to validate the presented certificate. The dissemination of the trusted root certificate and the maintenance of security and therefore trust along the certificate chain is the greatest weakness and complexity of any hierarchical PKI and the major factor against deployment of technologies utilising mutual PKI authentication.

4.3.2 Authentication in WLANs

As described in Chapter 2 [section 2.2.7], authentication in IEEE 802.11 WLANs is conducted using either of two methods: Open System Authentication, whereby a STA identifies itself and that identity is accepted without challenge by the receiver; or Shared Key Authentication, where a STA using the WEP protocol encrypts a challenge from the receiver in order to validate its WEP credentials.

Shared Key Authentication is only available where all STAs are using the insecure WEP protocol. All other Wireless networks, including Robust Security Networks (RSN), use Open System Authentication, that is non-validated acceptance of the initial identification — subsequently validated during the association phase.

Authentication allows a STA to move from State 1 to State 2 and therefore access the SS available from wireless STA to which it has authenticated. However, in order to access DSS, a station must both authenticate and associate with an AP providing DSS. It is during this association that the actual credentials of the requesting station of the participants may be verified.

4.4 Association

In order to access the DSS a station must associate with an AP providing these services. It is typically during this association phase that the identities of the participants are authenticated.

A STA requesting association selects an appropriate authentication and pairwise cipher suite from those advertised in the AP's cleartext Beacon frames or Probe Response frames and sends the details of its capabilities and the selected protection mechanisms in an unprotected Association Request of the format depicted in Figure 4.4.

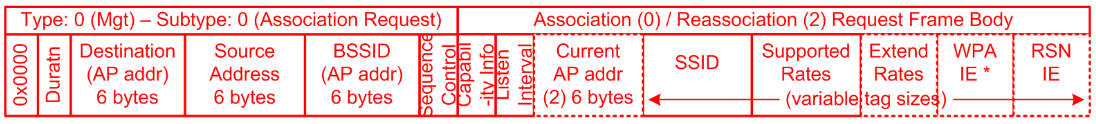


Figure 4.4: IEEE 802.11 Association Request Frame Format

The destination AP responds with an unprotected Association Response frame of the format depicted in Figure 4.5.

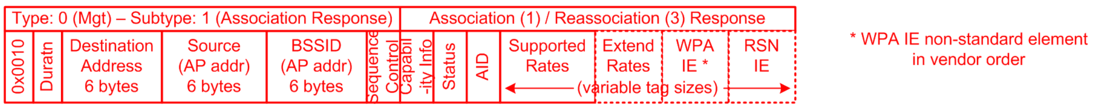

Type: 0 (Mgt) – Subtype: 1 (Association Response)
Association (1) / Reassociation (3) Response
0x0010
Duratn
Destination Address 6 bytes
Source (AP addr) 6 bytes
BSSID (AP addr) 6 bytes
Sequence Control
Capabil -ity Info
Status
AID
Supported Rates
Extend Rates
WPA IE *
RSN IE
(variable tag sizes)
* WPA IE non-standard element in vendor order

Figure 4.5: IEEE 802.11 Association Response Frame Format

Once association has completed, if the selected protections during the association phase indicated any form of RSNA, then the RSNA 4-Way Handshake immediately proceeds, accordingly authenticating the parties involved, as applicable. RSNAs may use either a Pre-Shared Key (PSK) or an IEEE 802.1X Extensible Authentication Protocol (EAP) method to derive the master keys.

LEAP

As already discussed in Chapter 2 [section 2.6.7], alternate solutions included the use of VPNs, SSL and Cisco's Lightweight Extensible Authentication Protocol (LEAP) — a proprietary (Cisco Systems) EAP method to deal with the existing WEP issues. It provided mutual authentication and dynamic WEP keys to resolve some of the initial problems identified in the WEP protocol. Even though

LEAP is not natively supported by any Microsoft OS, it uses the original Microsoft Challenge Handshake Authentication Protocol (MS-CHAP) and so user credentials are easily compromised [137].

The IEEE 802.11i MAC security enhancements introduce Robust Security Network Associations (RSNAs), as described in Chapter 3 [section 3.2], which provide for the use of IEEE 802.1X port-based Network Access Control (NAC) and its component Extensible Authentication Protocols (EAPs) as described in 2 [section 2.6.7]. It should be noted that while the IEEE 802.1X port-based NAC framework provides the basis for RSNAs, the strength of such associations relies heavily on the EAP method being used.

EAP-TLS

With the move towards IEEE 802.11i, and the release of the Wi-Fi Alliance's Wi-Fi Protected Access (WPA) certification, IEEE 802.1X supplicants where introduced with WPA associations, where EAP-TLS was the only mandatory EAP method. As such, EAP-TLS is supported by every vendor.

EAP-TLS is detailed in RFC 5216 [138]. It uses Transport Layer Security (TLS), to secure the authentication session. It provides mutual authentication using PKI certificates to secure communication to the RADIUS authentication server and therefore needs a pre-existing PKI in order to be deployed. While EAP-TLS provides the best security for wireless network associations, the necessary overhead of a pre-existing PKI prohibits its use in many situations outside of enterprise deployments. However for the enterprise with an existing PKI, EAP-TLS is the ideal choice of EAP method to secure wireless LANs.

EAP-TTLS

With the difficulties of using EAP-TLS, Funk Software and Certicom (now SafeNet) developed EAP-TTLS — Tunnelled Transport Layer Security — as a propriety EAP method, detailed in RFC 5281 [139], where the client *does not* need a PKI certificate. The server is authenticated using the server's PKI certificate and then a secure connection is used to tunnel the authentication of the client.

EAP-FAST

Cisco Systems replaced its insecure LEAP with EAP-FAST, detailed in RFC 4851 [140]. EAP-FAST also performs authentication via secure tunnelling, using a Protected Access Credential (PAC), with a fall back to TLS, to establish the tunnel. However difficulties lay in the secure distribution of the credentials.

PEAP

Cisco Systems, Microsoft and RSA developed the Protected Extensible Authentication Protocol (PEAP). PEAP has the same characteristics as EAP-TTLS, where server is authenticated using the server's PKI certificate and then a secure connection is used to tunnel the authentication of the client [137].

Even though EAP-TLS was previously the only EAP method mandatory for WPA certification, the Wi-Fi Alliance changed its certification scheme so that certification can be achieved with any of: EAP-TLS, EAP-TTLS/MSCHAPv2, PEAPv0/EAP-MSCHAPv2, PEAPv1/EAP-GTC, EAP-SIM [141] and, as of 19 May 2009, EAP-AKA and EAP-FAST [142]. Consumers buying retail equipment now have no way of determining which of the variations of the certification a piece of equipment has been assessed under.

Note that, irrespective of the EAP method(s) used in certification, vendors are free to include other EAP methods as they see fit. It has been the author's experience that a vast majority of offerings on store shelves include the insecure EAP-MD5 method. EAP was originally implemented for Point to Point Protocol (PPP) connections and one of the earliest EAP methods supported by major vendors such as Microsoft was EAP-MD5, as described in RFC 3748 [143]. The MD5 digest algorithm has numerous security issues including collisions and tools to find those collisions and EAP-MD5 does not provide mutual authentication, is not secure and should not be used.

4.5 Wireless Authentication Requirements

Wireless Local Area Networks (WLANs) have become ubiquitous in society today. The traveller expects a broadband connection in their hotel room, often also delivered via wireless means; wireless network access in the hotel common areas, including the swimming pool areas; wireless access in the airport lounge and in the numerous cafés and bookshops of the cites. Students expect uninterrupted wireless access, not only in auditoriums, but campus-wide wireless access across universities.

The corporate worker, using company SOE equipment, will have a mobile device, laptop, PDA, smartphone or the like, with either or both user and/or device credentials, already enrolled via the company registration system into the company unified directory.

In the past, before the deployment of RSNs, the corporate worker would connect their SOE equipment to the company's WEP-enabled WLAN, using the static pre-configured 104-bit hexadecimal WEP keys, generated by the company's system administrators at the beginning of the current millennium, and installed on every SOE device at commissioning ever since.

Once WEP's shortcomings became apparent, the corporate WLANs of the world boosted their network security, if not already, by enforcing VPN tunnelling over wireless links. Those more security conscious corporations ensured that all wireless links were treated with the mistrust they deserved by segregating all wireless activity into DMZs controlled by the company firewalls. In many cases, as the security was provided by the VPN, WEP was turned off to save the encryption overhead affecting the performance of the early wireless devices.

With the development and establishment of TSNs and RSNs, SOE devices now use WPA and WPA2 to authenticate via an EAP to an AP and RADIUS to a Network Access Control (NAC) Server which then interrogates the appropriate validation engines for the credentials supplied.

4.6 Threats to WLAN Infrastructure Association and Authentication

4.6.1 Passive Threats

Passive threats to WLANs involve the undetectable reception of the radiated broadcast wireless signals from any or all STAs of the WLAN, possibly at comparatively great distances from the sources, typically using inexpensive COTS equipment, by an assailant requiring minimal technological expertise, in order to gain the information in the transmitted messages or to gain information about the WLAN itself, often so as to mount more significant attacks.

Traffic Analysis

Traffic analysis is the simplest of the threats to wireless network security. Assuming the packets are encrypted and the assailant cannot decrypt the packets, then only the traffic volume, the number and size of the packets, the packet headers and possibly any packet trailers, including destination and source, exposed for analysis. [144] Traffic analysis can indicate network presence and significant events; AP vendors, types, MAC addresses and other beacon data; SSID (whether broadcast or not); STA locations; and many of the typical protocols being carried, unless packet size obfuscation is also being employed in some additional tunnelling protocol.

The encryption layer will determine amount of extra info. If the traffic is unencrypted, the assailant can read all of the data segments and reconstruct the entire communication session.

The only requirements for traffic analysis is a receiving antenna and a WLAN NIC in promiscuous mode, so as to process all traffic detected, whether it is addressed to the STA or not. The more sensitive the antenna, the greater this distance from which the traffic analysis can be performed.

Eavesdropping

Eavesdropping goes beyond simple traffic analysis and involves not only capturing packets, but also analysing the packet contents. This is trivial for unencrypted traffic, revealing all data transmitted, file transfers, web pages, emails, terminal services keystrokes and so forth. Eavesdropping not only compromises the confidentiality of the data in the wireless communications, but also provides the information required to more effectively undertake more damaging [144] active attacks, discussed below. Encrypted traffic, at any level, shall require the assailant to break the encryption being used in order to reconstruct the data segments.

Where WEP encryption has been used, it may take a number of hours to passively capture the 40,000 to 85,000 packets needed to break the encryption. However, if active methods are used, the encryption can be broken in a matter of minutes [11, 35, 145]. The numerous vulnerabilities of WEP have been discussed in the previous chapters, in section 2.6.6.

The only requirements for eavesdropping is a receiving antenna, a WLAN NIC in promiscuous mode and any of a wide variety of freely-available automated

WLAN packet capture and encryption breaking tools. Once again, the more sensitive the antenna, the greater this distance from which the eavesdropping can be performed.

4.6.2 Active Threats

Active threats to WLANs involve the detectable emission of signals in the WLAN spectra, including both omni-directional and narrow-beam transmissions to any or all STAs of the WLAN, possibly at comparatively great distances from the sources, but for a number of attacks necessarily physically between the legitimate STAs, often using inexpensive COTS equipment, by an assailant requiring only moderate technological expertise, in order to block, replay, insert, monitor or modify legitimate messages on the WLAN or to gain unauthorised access to the WLAN or any wired infrastructure beyond the WLAN.

Denial-of-Service

Denial-of-Service (DoS) is the simplest of the active WLAN threats to implement. The broadcast nature of radio communications leave them highly susceptible to any form of EM interference, both natural and man-made. The numerous DoS attacks on WLANs, including via unprotected Control frames and unprotected Management frames, have already been detailed in Chapter 2 [section 2.6.3].

Replay

Replay attacks record legitimate transmissions on the WLAN and retransmit the packets to provide false authentication credentials; to obtain duplicate actions from a service; or to elicit an encrypted response, so as to generate large volumes of encrypted traffic, in order to break an encryption scheme.

Packet Injection

Packet injection involves transmitting the assailants own packets, sometimes deliberately malformed, onto the WLAN, either broadcast to all STA or targeting a particular STA, such as an AP. This can be used to probe an AP for WLAN information; to generate an encrypted response to known plaintext, so as to aid in cryptanalysis; to inject a modified packet back into the WLAN; or to gain unauthorised access by joining an unprotected WLAN.

4.7 IEEE 802.11w Security

The IEEE 802.11i MAC security enhancements only provide protection for Data frames. It provides no protection for Management frames or Control frames. The shortfalls in security by not protecting these Management or Control frames, have already been described in the previous chapters. With other IEEE 802.11 Task Groups extending the use of Management frames to carry more and more sensitive information [146], such as *IEEE Std 802.11h-2003*, *IEEE Std 802.11e-2005*, *IEEE Std 802.11k-2008* and *IEEE Std 802.11r-2008*, the vulnerabilities of Management frames have become more and more important.

The proposed IEEE 802.11w amendment[2] for Protected Management Frames "to create enhancements to the IEEE 802.11 Medium Access Control layer to provide, as appropriate, mechanisms that enable data integrity, data origin authenticity, replay protection, and data confidentiality for selected IEEE 802.11 management frames" [147] only provides protection for a small but important subset of management frames — and only after the IEEE 802.11i key exchanges have completed. As such, this protection is only afforded after the IEEE 802.11i pairwise and groupwise keys, along with an additional IEEE 802.11w group key, have been set up and so, still exposes stations prior to key establishment.

The origin authenticity can only be guaranteed for unicast 'Robust' Management frames, including unicast Deauthenticate or Disassociate 'Robust' Management frames, but cannot be guaranteed for any sort of broadcast or multicast frames. Presumably, once the standard is ratified, compliant devices will ignore broadcast Deauthenticate or Disassociate messages.

The replay protection and MIC will be built into the 'Robust' Management frame's new IE and along with the overall message protection, once again, cannot provide protection prior to a STA's association and key exchanges. After the IEEE 802.11i pairwise and groupwise keys have been established, a third key, the Integrity GTK (IGTK) is also needed to provide the forgery protection (but not origin protection) for broadcast Robust Management frames.

The proposed amendment protects action management frames, deauthentication and disassociation frames. While this does close a large security hole in the malicious DoS attacks with forged disassociation and/or deauthentication of

[2]Readers please be aware, irrespective of any final acceptance or publication date on this thesis, this dissertation was written prior to the publication of any output from IEEE 802.11 Task Group W and thus may not necessarily reflect the current information in that respect.

legitimate STAs and now provides protection for the increasingly vital Action Management frames for established STAs, it still leaves a large amount of unprotected management traffic. However this work offers no protection for STAs when joining the network or when authenticating or when associating prior to the key exchange in the MAC security enhancement. Moreover, this provides no protection for an access point that must still provide beacons and still accept and respond to probe requests, authentication requests and association requests from STAs that are joining the network.

There is still no protection for control frames and, while malicious acknowledgments will achieve little, forged CTS frames can still be used to deny service to STAs in a network and forged RTS frames will still successfully deny the entire network — and floods of RTS or CTS frames will continue the DoS as long as the flood continues.

Finally there exists no technology currently to protect the data link layer as a whole and thus we present such a proposal in the next chapter.

Chapter 5

Link Layer Security

This chapter draws on the security needs and limitations at the link layer for wireless communications from previous chapters and details the proposal for Wireless Link Security (WLS). The implications and effects of encrypting MAC addresses, for both friend and foe, are raised with respect to both directed and broadcast traffic. A number of alternative algorithms a then introduced to allow for possible operational necessities or to simply increase speed. The protocol design is discussed, including the handling of the mutable fields, possible processing by various hardware components, the whitening functions and key establishment needs, including unicast, multicast and other broadcast requirements and finally, the deployment needs and the experiments required to determine the viability of such a scheme are detailed.

5.1 MAC Security

While many protocols encrypt MAC PDU (MPDU) payloads, such as the entire MAC SDU (MSDU), or higher-layer PDUs or their respective payloads [14–21], as depicted in Figure 5.1, and often also include a message integrity check (MIC) protecting the MAC headers (and any encryption headers) themselves [15], as well as frame source authenticity [19,20], the author has been unable to locate any examples of cryptographic confidentiality protection of link layer communications that include the MAC layer headers.

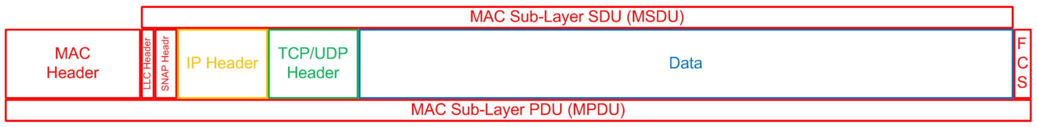


Figure 5.1: IEEE 802.11 MAC Frame Format

It is noted that many protocols also include copies of the MAC addresses and other MAC header data within the encrypted payload, such as MACsec [20] (also referred to by the project name LinkSec), but the data link layer cleartext headers are still transmitted in these cases. As such, these schemes may be able to protect the integrity of these headers, but cannot protect their confidentiality. While tunnelling protocols provide confidentiality of the tunnelled addresses, the link layer data of the tunnel itself remains exposed. There also exist schemes involving encrypted MAC addresses for the purposes of cataloguing or registration, such as in Yang and Liu [22], but these are not designed for communications protection.

TinySec [148] is a link layer security architecture for wireless sensor networks developed at the University of California, Berkeley. Being designed for sensor networks, TinySec requires a very small footprint, both in terms of packet size and computational complexity. As such, TinySec presents a very lightweight security architecture. Its designers made an observation that about 50–80% (sources 69.86%(2001)–61.60%(2004) [149]) of 802.11 networks operated in the clear, without any cryptographic protection whatsoever [148].

5.2 Wireless LAN Link Layer Security (WLS)

The author proposes a system to enhance the security of WLANs at the link layer, going beyond the MAC Security Enhancements of *IEEE Std 802.11i* [12] and the as yet incipient details of the Protected Management Frames of the coming amendment from IEEE P802.11w, to protect the entire data link layer, not just its payload, by encrypting the entire MAC frame (except the FCS) including the MAC header, shown in Figure 5.2.

This gives the potential for wholly encrypted transmissions at the link layer. Only the preamble has to be valid, which, being fixed, contains no information and the FCS, calculated across the encrypted frame, being entirely redundant over already encrypted data, also provides no information to an adversary.

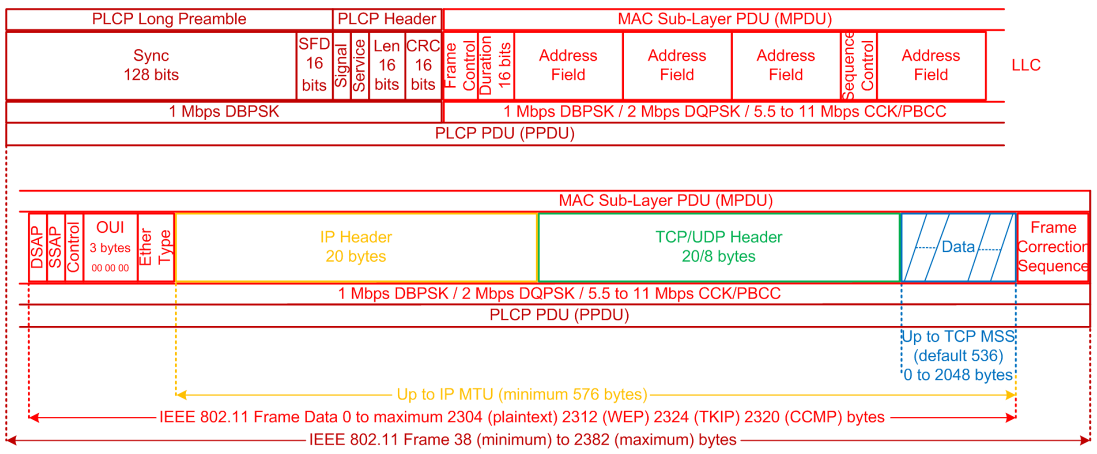

PLCP Long Preamble
PLCP Header
MAC Sub-Layer PDU (MPDU)
Sync 128 bits
SFD 16 bits
Signal
Service
Len 16 bits
CRC 16 bits
Frame Control
Duration 16 bits
Address Field
Address Field
Address Field
Sequence Control
Address Field
LLC
1 Mbps DBPSK
1 Mbps DBPSK / 2 Mbps DQPSK / 5.5 to 11 Mbps CCK/PBCC
PLCP PDU (PPDU)
MAC Sub-Layer PDU (MPDU)
DSAP
SSAP
Control
OUI 3 bytes 00 00 00
Ether Type
IP Header 20 bytes
TCP/UDP Header 20/8 bytes
Data
Frame Correction Sequence
1 Mbps DBPSK / 2 Mbps DQPSK / 5.5 to 11 Mbps CCK/PBCC
PLCP PDU (PPDU)
Up to TCP MSS (default 536) 0 to 2048 bytes
Up to IP MTU (minimum 576 bytes)
IEEE 802.11 Frame Data 0 to maximum 2304 (plaintext) 2312 (WEP) 2324 (TKIP) 2320 (CCMP) bytes
IEEE 802.11 Frame 38 (minimum) to 2382 (maximum) bytes

Figure 5.2: IEEE 802.11 Frame Detail

5.2.1 Encrypted MAC Addresses

Encrypting the MAC headers immediately conjures up images of lost frames with unintelligible addressing information, unable to get to the intended destination. However, link layer IEEE 802.11 networking, like IEEE 802.3 networking, uses a broadcast medium. Thus all frames are delivered to all possible destinations within range, including, in the case of the wireless medium (WM), any malicious or otherwise unwanted destinations also within range. It is then up to the receiving STA (STA) to decide whether or not the frame is intended for that particular STA.

Encrypting the MAC headers not only provides for confidentiality of the entire MPDU, but also aids in obscuring traffic patterns and hinders traffic analysis. As the traffic is broadcast to all STA in range, an adversary will be unable to trivially determine which STA was the intended recipient. Instead, an adversary will need to employ more advanced STA response detection techniques, such as correlating any individual return acknowledgment-only-sized frames or other STA response characteristic analysis. The employment of MPDU Aggregation and Block Acknowledgement will also further frustrate such analysis.

The basis of this proposal is that only a STA that has the correct key to decrypt the first block of the received frame can determine the addressing information and thus match the destination address to its own MAC address. A STA that does not have the correct key is not only unable to match the destination address, but is unable to determine which other STA the frame was intended for, nor even which STA the frame is from or any other frame details. To all other STA, the frame addressing appears to be corrupt.

5.2.2 Link Layer Needs

In order to be able to discuss how this proposal may be designed or implemented on IEEE 802.11 networks, it is necessary to first discuss the detailed link layer protocols and framing components being used. IEEE 802 networking operates at the Data Link and Physical layers of the ISO OSI Model.

Unlike IEEE 802.3 networking, where the broadcast signals are for all intents and purposes restricted to some wired or guided medium, IEEE 802.11 signals are typically broadcast omnidirectionally beyond the jurisdiction and control of the network owner and encompass a superset of the threats to IEEE 802.3 networks. As such, the checks and balances and cost-efficiency tradeoffs relevant to other networks are different in IEEE 802.11 networks.

Previous decisions on the relative merits of functionality and features versus complexity and computational expense, reasoned with regard to IEEE 802.3 networking in particular or having to apply equally to all forms of IEEE 802 networking, should be tempered with the particular needs of wireless network security when applied to IEEE 802.11 networks. For example, although the IEEE 802.1 Link Security Study Group (LinkSec) specifically ruled out MAC address encryption and identity hiding during the development of IEEE Std 802.1AE–2006 (MACsec) [20], this standard is designed to apply to all IEEE MAC layers and "didn't specifically account for wireless"[1] [19].

5.2.3 The Logical Link Control Sub-Layer

The Logical Link Control (LLC) sub-layer uses the IEEE 802.2 protocol, as do all IEEE 802 LAN protocols including the Ethernet-like IEEE 802.3 CSMA/CD, but not the (non-IEEE) Ethernet II protocol itself.

5.2.4 The Physical Layer Convergence Protocol

The Physical Layer Convergence Protocol (PLCP) adds the frame preamble to synchronise the radios, finishing with the Start of Frame Delimiter (SFD) and the PLCP header.

[1] "Last thing anyone wants is to have more to worry about in wireless." [19]

IEEE 802.11 FHSS

The IEEE 802.11 FHSS preamble consists of 80 bits of alternating zeros and ones, 0101...01. The fixed 16-bit SFD is 0000 1100 1011 1101. This is followed by the PLCP header. The 32-bit IEEE 802.11 FHSS PLCP header consists of the 12-bit PSDU Length Word (PLW), the 4-bit PLCP Signaling Field (PSF) encoding the payload transmission rate and the Header Error Check (HEC) 16-bit CRC.

IEEE 802.11 DSSS

The IEEE 802.11 DSSS preamble consists of 128 bits of all ones, 1111...11, subsequently scrambled by the DSSS process, along with the rest of the frame. The fixed 16-bit SFD is 0000 0101 1100 1111, which is different from the FHSS SFD. This is followed by the PLCP header. The 48-bit IEEE 802.11 DSSS PLCP header consists of the 8-bit Signal field encoding the payload transmission rate — 0000 1010 for 1 Mbps or 0001 0100 for 2 Mbps — the all-zero 8-bit Service field, 00000000, for future use, the 16-bit Length field set to the number of microseconds required to transmit the frame as an unsigned 16-bit integer — transmitted least significant bit to most significant bit [150] — and the 16-bit CRC.

IEEE 802.11b HR/DSSS Long PLCP Format

The IEEE 802.11b DSSS long preamble consists of 128 bits of all ones, 1111...11, subsequently scrambled by the DSSS process, along with the rest of the frame. The long format 16-bit SFD is 1111 0011 1010 0000 — transmitted least significant bit to most significant bit [150]. This is followed by the PLCP header. The 48-bit IEEE 802.11b HR/DSSS long PLCP header consists of the 8-bit Signal field encoding the payload transmission rate — 0000 1010 for 1 Mbps, 0001 0100 for 2 Mbps, 0011 0111 for 5.5 Mbps or 0110 1110 for 11 Mbps — the 8-bit Service field flagging clocks and modulation and extending the Length field to 17 bits, the 16-bit Length field set to the number of microseconds required to transmit the frame as an unsigned integer and the 16-bit CRC. The long header is transmitted at 1 Mbps using DBPSK.

IEEE 802.11b HR/DSSS Short PLCP Format

The IEEE 802.11b DSSS short preamble consists of 56 bits of all zeros, 0000...00, subsequently scrambled by the DSSS process, along with the rest of the frame. The short format 16-bit SFD is 0000 0101 1100 1111 — the reverse of the long format. This is followed by the PLCP header. The 48-bit IEEE 802.11b HR/DSSS short PLCP header consists of the 8-bit Signal field encoding the payload transmission rate — but only 2 Mbps, 5.5 Mbps, and 11 Mbps are defined [150] — the 8-bit Service field flagging clocks and modulation and extending the Length field to 17 bits, the 16-bit Length field set to the number of microseconds required to transmit the frame as an unsigned integer and the 16-bit CRC — and the 16-bit CRC. The short header is transmitted at 2 Mbps using DQPSK.

IEEE 802.11a OFDM PLCP Format

IEEE 802.11a OFDM uses operating channels with 52 sub-carriers. Of these, 4 sub-carriers are used as pilot carriers and the remaining 48 are used to carry the data. The IEEE 802.11a OFDM preamble consists of 12 OFDM symbols over 16 µs — one window with ten repetitions 0.8 µs "short training sequence" [71] symbols with no guard interval, then one window with a 1.6 µs guard interval and two 3.2 µs "long training sequence" symbols. This is followed by the PLCP header. The IEEE 802.11a OFDM PLCP header consists of one single OFDM symbol providing the 24-bit Signal field, transmitted using DBPSK with a convolution code at a rate of $R = \frac{1}{2}$(one data bit for every two code bits); and a 16-bit Service field in the first data symbol, transmitted with the rest of the data at the payload transmission rate. The 24-bit Signal field contains the 4-bit Rate, encoding the payload transmission rate; one reserved bit, set to zero; the 12-bit Length field, encoding (least-significant bit to most-significant bit) the number of bytes in the embedded MAC frame; a positive parity (even parity) bit for bits 0–16 [71] and all-zero six Signal Tail bits. The 16-bit Service field contains six zero bits for Scrambler Initialisation and nine reserved (zero) bits for future use.

After the message end the IEEE 802.11a OFDM PLCP adds a Tail field of six zeros to bring the convolution encoder back to its "zero" state [71]. After encoding, the scrambled 6-bit tail is replaced with six non-scrambled zero bits to also bring the receiver's convolution decoder back to its "zero" state. Finally the number of Data bits needs to be a multiple of the Number of data Bits Per Symbol, $N_{NBPS} =$ 48, 96, 192 or 288 bits. The message is padded with zero bits, which are subsequently scrambled with the rest of the Data bits. Pad bits are removed before convolution decoding.

5.2.5 Distinction

This proposal should not be confused with the Fortress Technologies "Wireless Link Layer Security (wLLS)" [151] which was released in 2001 as a WEP replacement technology. Nor should this be confused with "WLLs", Wireless Local Loops [152], a solution for the replacement of last-mile wired local loops for provision of telecommunication services to subscribers in remote or difficult areas. For this reason, the acronym used here is deliberately kept simple, "WLS", for "Wireless Link Security".

5.2.6 Defeating Sniffers

The effect of this proposal with complete link layer encryption is that any party monitoring the wireless medium will see the frame preamble for an IEEE 802.11 frame, either short or long as applicable, and the PLCP header, however all data following the PLCP, being encrypted, shall appear unintelligible. This not only provides for confidentiality of the PSDU itself but also provides confidentiality of the identities of the transmitting and receiving STAs.

Any monitoring party shall be unable to determine which receiver the frame is intended for, the MAC headers, or the frame content. An adversary using packet capture tools such as Kismet, Wireshark, wepcrack or the like, will be unable to determine the destination STA for any frame transmitted in the broadcast medium, nor can the source STA address be determined from the MAC headers where the source address has been encrypted. All current wireless sniffers and wireless cracking programs and devices will identify the frame as corrupt (the FCS will be valid, but the protocol version will be invalid) or otherwise report nonsense data.

Even where radio spectral analysis equipment is used to determine which STA is transmitting an encrypted frame, using signal strength correlation or direction triangulation, an adversary is still unable to determine which STA the frame is intended for since the broadcast medium carries the frame to every STA in radio range. This is more effective if the wireless links utilise Block Acknowledgements, or even unacknowledged multicast frames, rather than positively acknowledged directed frames, where the acknowledgement may reveal the true recipient of the frame.

Where the frames are encrypted with a large block cipher even the true length of the transmitted frame is hidden from an intruder. However the larger the block cipher used the greater the encryption overhead in determining whether an individual frame is valid at the legitimate receiver.

5.2.7 Determining the Source

With sufficient resources the transmitting STA at any particular moment can be determined by signal strength correlation or direction triangulation, however this can no longer be gleaned from a simple packet capture and if all communications are encrypted and the STAs never present their MAC address, or at least never the present the MAC address used for 802.11, then this will not even provide a vector for a known-plaintext attack. At the very best there could be a 32-bits-of-unknown-plaintext-but-known-to-be-the-same attack. Obviously, the fewer the number of participating STAs and the longer the key rotation time, the greater the viability of any such attacks. The author leaves this analysis to the cryptographic experts.

This introduces a significant overhead in that all received frames must now have at least their first block decrypted before determining if the frame is even intended for the receiving STA. Every STA must decrypt the first block of every frame transmitted by any other STA. In infrastructure mode, this is a considerable overhead for the individual STA as they must not only decrypt the frames from the AP, but must also decrypt the first block of every frame from every other STA, intended for the AP, just to determine the frame is not for the STA. This overhead is amplified for the AP in infrastructure mode (and for all STA in ad hoc WLANs), where multiple pairwise keys are held, the first block of every frame must be decrypted with each of the successive keys to see which if any of the keys successfully decrypt the destination address to match that of the receiving STA.

5.3 Partial WLS

With full link layer encryption creating such a large overhead this introduces the possibility of leaving the source address (or some equivalent index) unencrypted so that a STA holding multiple pairwise keys can identify which key should be used to attempt the decryption of the frame. Although this now provides a source address to an adversary, a determined adversary can already determine the transmitting STA via correlated signal strength or direction triangulation. By providing the source address unencrypted this significantly reduces the overhead for those STAs communicating with multiple peers, such as an AP in infrastructure mode or any STA in ad hoc mode. An adversary spoofing the source address of a legitimate STA is still unable to provide a frame that will decrypt correctly as the adversary does not have the key to encrypt the remaining components of the frame correctly.

This now gives us two modes of link layer security, full Wireless LAN Link Layer Security (WLS) involving the encryption of the entire layer 2 MPDU and Partial Wireless LAN Link Layer Security (PWLS) where the source address of the transmitting STA is left in the clear.

5.4 Faster Algorithms

Another possibility is to encrypt the MAC addresses only with a single group key shared by all legitimate STAs, so that only legitimate participants of the network can determine which STAs each message has come from and is intended for. An intruder not holding the group keys will be unable to determine the legitimate addresses of either transmitting or receiving STAs. However this method does not protect individual STAs from malicious activity by legitimate parties within the network.

A further amendment to this is to encrypt the *source* MAC addresses only with a single group key shared by all legitimate STAs (*destination* MAC addresses remain encrypted with the appropriate pairwise key), so that only legitimate participants of the network can determine which STA each message has come from and must decrypt the first block if it has a matching pairwise key to see if it matches the destination address. Once again, an intruder not holding the group keys will be unable to determine the legitimate addresses of either transmitting or receiving STAs. In this case, malicious legitimate parties within the network can learn the source addresses of all other participants, but remain unable to spoof destination addresses, while spoofed source addresses frames will fail to decrypt.

5.5 Protocol Design

5.5.1 Immutable Fields

The frame checksum (FCS) is calculated over the entire frame including source and destination addresses. This value is determined by the hardware at transmit time and in the case of WLS includes the encrypted source and destination addresses, whereas in PWLS shall be calculated using the encrypted destination address and unencrypted source address. This will not affect the receiving hardware under either method and the receiver shall be able to correctly determine the FCS appropriate for the received frame.

Similarly the framing of the PSDU into a PPDU, may well occur in hardware, however as this has no information that cannot already be determined by analysis of the frame as a whole, there is no value in encrypting this and, indeed, to encrypt the PLCP preamble and SF delimiter simply provides more data for known-plaintext attacks on the system.

Where the various fields of the MAC headers are generated by hardware, prototyping of these designs will require the construction of the entire PSDU in software, to then be encrypted, so that the various header components can be individually provided already encrypted to the existing MAC hardware via appropriate calls to the wireless network interface device drivers, where such low-level access is available. This is most likely to be possible using popular open-source drivers for highly-configurable NICs, such as those using Atheros chipsets.

The author regularly builds such drivers, however prototyping modifications to the open-source code-base is complex at best, often precarious, typically leads to failure, and is beyond the scope of this work.

Note the proposed encryption occurs at the MAC layer, before the physical layer processes. As such, the physical layer whitening for the signal occurs after the assembly of the frame with its encrypted payload, and so is independent of the encryption process. The physical layer whitening is only for transmission functionality and has no effect on the encryption or its relative strength or any weaknesses.

5.5.2 Duration and NAV

It should be noted that any of the methods as already proposed here will render the Duration field unreadable to all but the destination STA. However, this field, when used in its role to support the virtual carrier sense mechanism, is intended to be used by every STA *except* the destination, to update the Network Allocation Vector (NAV), as descibed in Chapter 4.

As they currently stand, these proposals will therefore have some detrimental effect on the virtual carrier sense mechanism, which will come into effect wherever there are 'hidden nodes' or other cases of signal loss within the network. This work does not investigate the extent of any such effect, however, it would be prudent for any further design to consider the value of *not* encrypting this field when it contains a Duration value (0–32767).

In this manner, if the field is to be encrypted for ID or reserved use, only bits 0-14 should be encrypted and bit 15 always set so that the encrypted value cannot fall within the range of a Duration value.

This approach has merit in that the Duration value, when set as such, can be determined from the frame size and bit rates and so form known plaintext within the header. As such when this field contains a Duration value it would be better not to include such known plaintext in the encryption.

5.5.3 Key Establishment

It is possible to establish the pairwise keys through public key encryption techniques using the same certificates that would be present for some of the existing technologies as defined by IEEE 802.11i standards and implemented under the Wi-Fi Alliance's WPA and WPA2 certifications, such as EAP-TLS, as discussed in Chapter 4.

The use of public key cryptography for key establishment introduces the risk of denial of service attacks due to the expensive computational overheads of public key cryptography. This can be mitigated using staged authentication involving less intensive cryptographic techniques as the identity of the peers are established finally leading to full public key cryptographic authentication. Such techniques are beyond the scope of this work.

5.5.4 Multicast and Broadcast

Another issue is multicast and broadcast frames where a single packet must be delivered to multiple addresses. This will require the use of a broadcast key known to all legitimate parties — however this is not the same as the IEEE 802.11i groupwise key, nor the IEEE 802.11w Integrity GTK (IGTK).

The draft of the IEEE 802.11w standard, which adds security provisions to a very limited set of management frames[2], does rely on the IEEE 802.11i keys, and as such provides no protection before this key exchange has occurred. In addition, the IEEE 802.11w standard introduces its own groupwise key, the IGTK needed to provide the forgery protection for broadcast Robust Management frames.

For our proposals here, the intention is to use pre-existing key material such as from a public key infrastructure (PKI), to authenticate and support key exchanges to set up pairwise and groupwise WLS keys in the manner of an EAP-TLS exchange, as a precursor or replacement of the usual Probe Request-Response exchange. This proposal thus necessarily precludes the much of the general-purpose nature of WLANs, as the legitimate STA must already have appropriate PKI credentials to participate in the network. However, such restrictions do not pose an unreasonable requirement given the level of protection being afforded by this proposal.

5.6 Viability of the Thesis

Having now established this proposal, we need to determine its possible viability. While the need for existing key distribution methods does limit the application of these new protocols, of more concern is the effects of this additional layer of encryption delaying medium access and thereby damaging network throughput. As the encryption is being applied at the MAC sub-layer, there is more potential for encryption delays to greatly affect access to the medium, with devastating consequences for network throughput — and therefore the protocol's viability.

This research needs to determine what the realistic penalty is as a result of these encryption delays; and whether the proposed faster algorithms, including PWLS, are needed to make the system viable, or indeed, whether it is worth

[2]Readers please be aware, irrespective of any final acceptance or publication date on this thesis, this dissertation was written prior to the publication of any output from IEEE 802.11 Task Group W and thus may not necessarily reflect the current information in that respect.

deploying WLS at all.

The critical characteristics that needed to be determined include the effects of the encryption penalties on the protocols, individual STAs and the network as a whole; and how these vary with the mobility of the STAs, the range of the STAs, and indeed, the overall size of the network in physical numbers of STAs.

The following chapters will detail the tests undertaken to determine these factors.

Chapter 6

Developing the Tools

The proposal in the previous chapter has identified a significant risk in the potential for encryption delays to greatly affect access to the medium, with severe impacts on network throughput. This work will now proceed to determine what the realistic penalty is as a result of these encryption delays and whether WLS is a viable proposition, or whether the proposed faster algorithms, including PWLS, are needed to make the system viable.

The author has conducted numerous simulations, with both small and large numbers of participating STAs, in the pursuit of empirical data to test the viability of these proposals. This chapter details the process of developing the various tools needed for all stages of simulation, analysis and reporting. This includes the selection of the tools themselves and platforms to be used to provide them, as well as the configurations and validation of these tools against expected results, both in theory from the IEEE Std 802.11 series and in practice from other research data [153]. This is then followed by the actual conduct of the tests and the analysis of the results in the next chapter.

A number of different tools were utilised for this purpose: *ns-2* [154] to simulate the operation of an IEEE 802.11 network under the DCF; recording of traffic sink data to determine throughput; trace parsing scripts to analyse the simulated network performance; spreadsheets to determine and visualise backoff periods and the choice of backoff slots by contending STAs; and *xgraph*[1] [155] and *gnuplot* [156] to visualise the results.

[1]This is David Harrison's BSD *xgraph*, not Carl Hein's proprietary *XGRAPH*.

6.1 ns-2

ns-2 is a network simulator born out of LBNL[2]'s Network Simulator, *ns* (now also known as "*ns-1*"), itself developed from Srinivasan Keshav's *REAL* (REalistic And Large) [157] simulator, which in turn used the the *NEST* (NEtwork Simulation Testbed) [158] toolkit from Columbia University. *ns-2* was developed by the DARPA-funded Virtual InterNetwork Testbed (VINT) research project in the late 1990's. The VINT project involved the University of Southern California's (USC) Information Sciences Institute[3], the then Xerox PARC[4], LBNL and UCB[5]. *ns-2* has been enhanced over the years by the USC Information Sciences Institute's DARPA-funded Simulation Augmented by Measurement and Analysis for Networks (SAMAN) and NSF[6]-funded Collaborative Simulation for Education and Research (CONSER) projects, the International Computer Science Institute's (ICSI) Center for Internet Research (ICIR)[7], as well as a large and active research community, including Sun Microsystems, the UCB Daedalus (BARWAN[8]) project and the *Monarch Project's Wireless and Mobility Extensions to ns* [25] from CMU[9] and Rice[10] University.

ns-2 is written in *C++* and uses an *OTcl* (Object-oriented Tool Command Language) interface to set up the simulation.

An NSF CISE[11] Computing Research Infrastructure (CRI) project is currently developing *ns-3*, a re-design of *ns*, written in *C++* and *Python*. *ns-3* was conceived in 2006 to resolve perceived issues [159] in *ns-2*, whose formal development funding finished in 2004, such as issues with simulating wireless networks, many of which have since been successfully resolved in *ns-2*. *ns-3* was still under heavy development at the time of writing this thesis [160].

ns-2 was chosen for this work because it is a mature, well-developed simula-

[2]Lawrence Berkeley National Laboratory, previously "LBL" (Lawrence Berkeley Laboratory), commonly "Berkeley Lab", operated by the University of California, Berkeley.

[3]The USC Information Sciences Institute's acronym is not used in this dissertation. All occurrences of "ISI" in this thesis refers to the Queensland University of Technology's Information Security Institute — this author's affiliation.

[4]Now just Palo Alto Research Center Incorporated (PARC), since 2002.

[5]The University of California, Berkeley.

[6]The U.S. National Science Foundation.

[7]The ICIR was formerly the AT&T Center for Internet Research at ICSI (ACIRI).

[8]The Bay Area Research Wireless Access Network.

[9]Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

[10]Rice University, Houston, Texas, USA.

[11]The U.S. National Science Foundation Directorate of Computer and Information Science and Engineering (CISE).

tion engine and well-regarded by the research community. As such, *nam* is then the logical tool of choice for animating packet trace data from the simulations.

Version *ns-2.32* was initially chosen, but was immediately replaced by *ns-2.33*, being the most current stable release, as at 31st March 2008. The "*ns-allinone-2.29.3*" version was not used as the *ns-2.3x* versions contained required wireless additions for this work.

Tcl/Tk version 8.4.14, a version verified to work for current *ns-2* sources, was the ultimate choice for these tools, combined with *OTcl* version *otcl-1.13*, *TclCL* version 1.18, for "Tcl with CLasses", *nam* version 1.13 and David Harrison's *xgraph-12.1* were used.

See the installation and testing of the *ns-2* network simulator and associated tools described in detail in Appendix A.

6.1.1 Platform Selection

The *ns-2* network simulator and *nam* network animator were installed on a number of machines over the course of this work. These included *Intel* 32-bit and *AMD*[12] 64-bit platforms, running *Fedora 7* and *Fedora 9* operating systems (OS), ultimately on a *Fedora 9* 32-bit single-core *VMware* instance hosted in a *Microsoft Windows XP* 32-bit OS on *Intel* 64-bit dual-core hardware.

The preparation of the platforms and the installation and testing of the associated tools are described in detail in Appendix A.

Unix[13]-like platforms were chosen because *ns-2* is developed on *Unix* or Unix-like platforms and although also ported to *Microsoft Windows* OS variants, has far better community support and functionality on Unix-like platforms. Of these, *Linux* distributions are the most common Unix-like OS variants with free, or at least open-source, licensing allowing use, investigation and modification of the code without restriction.

This author's choice of the *Red Hat / CentOS / Fedora* OS family was purely the result of a long history of using Red Hat products and the commercial realities in the author's experience of particular enterprise products that are only supported on *SunOS/Solaris* or *Red Hat Enterprise Linux* (RHEL). A reader wishing to replicate these experiments may well prefer a *Debian* distribution or

[12]Advanced Micro Devices, Inc.

[13]The *Unix* name uses lower-case and is not an acronym, Bell Labs started using and eventually trademarked *UNIX* after Dennis Ritchie and Ken Thompson's 1974 paper, *The UNIX Time-Sharing System* [161], where they used UNIX in *troff*'s new 'small caps' capability.

a derivative, such as *Ubuntu*, or some other Unix-like OS, since command-line and basic GUI operations are the same on any distribution. Appendix A describes the set-up on a Fedora distribution and may not involve identical commands and configuration on other systems.

Initially, the *Fedora 7* OS was chosen both for its currency over the very stable *CentOS* (a re-compiled community version of RHEL) and its stability over the latest release, at that time, of *Fedora 8* (a *Red Hat* sponsored community *Linux* project). As *ns-2* is compiled against the current running kernel, the frequency of kernel upgrades has considerable impact on research activity involving the simulator. By not being the latest release, at that time, *Fedora 7* was likely to have less frequent kernel upgrades and thus provide a longer mean time between full compilations of the software. However, as the research progressed, *Fedora 7* fell too far behind in development, presenting unsupported packages and a general security risk and so it was decided to move to *Fedora 9*.

6.1.2 Preparation and Installation of the Simulator

The preparation of the platforms and the installation and testing of the *ns-2* network simulator and associated tools are described in detail in Appendix A.

This then gave a working installation of the *ns-2* simulator and *nam* animator. A number of test simulations were then conducted, both to test the installation and to familiarise the author with the simulator.

6.2 Validating the Simulator

A number of procedures were undertaken to validate the operation of the simulator tool. Firstly, the actual build for the platform was tested as follows.

```
$ ns
% set ns [new Simulator]
_o4
% ^C
```

This shows that the simulator does run on the chosen platform. However, it does not provide any assurance as to the correct operation of the simulator. The *ns-2* simulator package comes with a suite of validation tests to verify the build of the code-base. This test suite is generated with the compile of the rest of the source and is available in the *~ns/tcl/test* directory. This test suite includes the original *ns-2* validation tests detailed by Floyd [162]. These tests can be performed by calling the "./tcl/test/test-all-simple" script (this replaces the previous "./test-all" script in the *~ns* directory). However, these original tests are only a subset of the full suite. All of the tests can be performed by calling the "./validate" script in the *~ns* directory. Typing this command provides hours of scrolling output of the form shown below.

```
Running test sack1c:
../../ns test-suite-sack.tcl sack1c QUIET
Guide: SACK TCP, many packet drops, window=27
Test output agrees with reference output
Running test sack3:
../../ns test-suite-sack.tcl sack3 QUIET
Guide: SACK TCP, drops from a small window
Test output agrees with reference output
Running test sack5:
../../ns test-suite-sack.tcl sack5 QUIET
Guide: SACK TCP, many drops, without maxburst
Test output agrees with reference output
```

and so forth, totalling 6,646 lines of output, with the final summary shown at the end:

```
Running test aloha.collisions:
../../ns test-suite-satellite.tcl aloha.collisions QUIET
Test output agrees with reference output
Running test mixed.legacy:
../../ns test-suite-satellite.tcl mixed.legacy QUIET
Test output agrees with reference output
All test output agrees with reference output.
Fri Mar 27 02:34:26 EST 2009
These messages are NOT errors and can be ignored:
    warning: using backward compatibility mode
    This test is not implemented in backward compatibility mode

validate overall report: all tests passed
```

This full test suite was run every time the simulator was modified and the final validation test results for the modified simulator are archived and stored with the rest of the source, scripts, and result data for this work.

Having validated the correct operation of the *ns-2* simulator tool, the work proceeded with determining the required configuration of the simulator need to conduct the testing of the proposed protocols.

Initial investigations with *ns-2* showed discrepancies between nominal data rates and the actual throughput observed. A series of simulations were conducted to determine the cause of these anomalies.

6.3 Graphing Tools

Both the VINT release of *xgraph* [155] and the current released version of *gnuplot* (4.2.5) [156] were used to visualise the results from the simulations.

David Harrison's *xgraph* is a versatile graphing tool optimised for use with the X Window System (X11) being used on the simulation instances. All simulations were configured to automatically call *xgraph* on completion to provide an immediate display of the simulation results.

However, as the work progressed, the simulation results became more complex and the representation of the results more detailed, requiring careful configuration of the data representations in order to provide meaningful output. For these purposes, *gnuplot* proved to be a more flexible and versatile application to manipulate the representation of the simulation results. The final representations of the data presented in this document were produced using the *gnuplot* software, configured to provide the optimal resolution in these printed pages, stored and reproduced at 200 pixels per inch, using varying markers to compensate where black and white reproductions are made of the original full colour images.

6.4 Initial Configuration

This work does not detail the general configuration, use and operation of the *ns-2* simulator tool. There are a myriad of resources on that subject and the remainder of this dissertation assumes the reader is at least familiar with the general concepts of running *ns-2* simulations.

Initially, all configuration was performed via Tcl scripts setting parameters, initialising and running the *ns-2* simulator. All of the scripts used a number of common features. In all cases, the *C++* wireless node objects were instantiated with bindings to a Tcl array called `node_`; UDP and TCP agents and applicable traffic sinks were bound to `udp_`, `tcp_` and `sink_` arrays in the Tcl scripts; and any Constant Bit Rate (CBR) UDP applications or FTP TCP applications were bound to `cbr_` or `ftp_` Tcl arrays. All simulations used the standard WirelessChannel, with the TwoRayGround radio-propagation model and the OmniAntenna model. All simulations used the standard *ns-2* `Phy/WirelessPhy` physical layer and `LL` "link layer" [163] (LLC). Initially, the MAC type was also the standard CMU/Rice `Mac/802_11` enhancement [25], now included in the base *ns-2* distribution.

Figure 6.1 shows the throughput for simplex Constant Bit Rate (CBR) UDP messages being generated at 11 Mbps on `node_(1)` and transmitted to `node_(0)` over an IEEE 802.11 WLAN, starting at time 1.0. The relevant settings from the simulation configuration Tcl script, *drtest1.tcl*, are:

```
set val(chan)     Channel/WirelessChannel   ;# channel type
set val(prop)     Propagation/TwoRayGround  ;# radio-propagation model
set val(netif)    Phy/WirelessPhy           ;# network interface type
set val(mac)      Mac/802_11                ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue   ;# interface queue type
set val(ifqlen)   50                        ;# max packet in ifq
set val(ll)       LL                        ;# link layer type
set val(ant)      Antenna/OmniAntenna       ;# antenna model
set val(routing)  DSDV                      ;# DSDV or DSR or TORA
set val(nn)       2                         ;# number of mobilenodes
set val(x)        350                       ;# X dimension of the topo
set val(y)        350                       ;# Y dimension of the topo
set val(sc)       "drtest-2-1-scen"         ;# scene movement file
set val(cp)       "drtest-2-1-cbr-11"       ;# traffic pattern file
set val(ns)       5.0                       ;# node speed
set val(bps)      11.0                      ;# CBR bit rate in Mbps
set val(stop)     60.0                      ;# simulation time
```

The LLC processing delay for the encapsulation of the UDP datagram into the LLC PDU was set to (a nominal) 25 µs. The class LL simulates the LLC sub-layer and is coded in *C++* files *~ns2/mac/ll.h* and *~ns2/mac/ll.cc* and includes `delay_`, the link layer delay. This should be around 25 µs for wireless simulations [164].

```
LL set delay_ 25us
```

The channel type, radio-propagation model, network interface type, MAC type, interface queue type and size, link layer type, antenna model and ad-hoc routing type are all used in the creation of the wireless nodes. The x and y values set the size (in metres) of the boundaries of the wireless topography and nn is the number of mobile nodes. These are all utilised as follows.

```
set ns_ [new Simulator]
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
# Create God, the General Operations Director
set god_ [create-god $val(nn)]
# Create channels here instead of -channelType in node-config
set chan_1_ [new $val(chan)]
# Configure the mobilenodes and then create them
$ns_ node-config -adhocRouting $val(routing) \
                 -llType $val(ll) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -macType $val(mac) \
                 -phyType $val(netif) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -channel $chan_1_ \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace OFF \
                 -macTrace OFF \
                 -movementTrace OFF
#  Create the specified number of nodes [$val(nn)] and "attach" them
for {set i 0} {$i < $val(nn) } {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0  ;# disable random motion
}
# Load the node movement model
source $val(sc)
```

Here, the scene movement file used, *drtest-2-1-scen*, starts with two nodes at

1 m range until time 2.0, the nodes then move apart to a range of 300 m at 5 m/s. The radio range is 250 m, so the connection fails at the end of the simulation, as seen after 50 seconds of movement in Figure 6.1. The relevant settings in the scene movement file are:

```
$node_(0) set X_ 25.000000000000
$node_(0) set Y_ 175.000000000000
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 26.000000000000
$node_(1) set Y_ 175.000000000000
$node_(1) set Z_ 0.000000000000
$ns_ at 2.000000000000 "$node_(1) setdest 325.000000000000 175.00000000
0000 5.000000000000"
```

A traffic pattern file, *drtest-2-1-cbr-11*, was generated for these tests using the CMU/Rice *Monarch Project's Wireless and Mobility Extensions to ns-2* [25] *cbrgen.tcl* Tcl script, as follows.

```
ns cbrgen.tcl -type cbr -nn 2 -seed 1.0 -mc 1 -rate 11.0 > drtest-2-1-c
br-11
```

It was subsequently realised that the "rate" was the packet rate, not the data rate and so had to be corrected, as the above command results in 11 x 512-byte packets per second, a packet interval of $0.\dot{0}\dot{9}$ seconds, not 11.0 Mbps, requiring another 512-Byte packet every $0.000372\dot{3}\dot{6}$ seconds. Table 6.1 displays the formulae for setting packet rate or interval for various packet sizes to achieve a given Data Rate, R, in Mbps.

The CMU/Rice *cbrgen.tcl* tool produces unexpected node numbering. The `-nn` flag is intended to be the maximum number of mobile nodes, and the resulting output file lists the value given at `-nn number` as "nodes: `number`", `number` is used as the highest node identifier and nodes are numbered from '0', so simulations must account for node numbers in the range 0–`number`, or `number`+1 nodes in total. This can be replicated with the command `ns cbrgen.tcl -type cbr -nn 50 -seed 0.1 -mc 50 -rate 2685.546875 > cbr-50-test`, which produces an output headed "`# nodes: 50, max conn: 50, send rate: 0.00037236363636363639, seed: 0.1`" with 39 actual connections from 29 sources, including both `node_(0)` and `node_(50)`.

Packet Size	Packet Rate (packets/second)	Packet Interval (seconds)
general formula	R x 1000000 / Packet Size	1 / (R x 1000000 / Packet Size) = Packet Size / (R x 1000000) = Packet Size / 1000000 / R
bits	R x 1,000,000	0.000001/R
bytes	R x 125,000	0.000008/R
512 B	R x 244.1406250	0.004096/R
1 KiB	R x 122.0703125	0.008192/R
2 KiB	R x 61.03515625	0.016384/R

Table 6.1: Calculations of packet rate and interval for various packet sizes, given the desired Data Rate, R, in Mbps.

The *cbrgen.tcl* tool generates a random number of connections, up to the maximum specified, between a random number of nodes, uses random start times and turns on random noise. This was not suitable for the highly-controlled tests needed at this stage, so the resultant output needed to be modified, as shown below.

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set sink_(0) [new Agent/LossMonitor]
$ns_ attach-agent $node_(1) $sink_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
# $cbr_(0) set interval_ 0.090909090909090912
$cbr_(0) set interval_ 0.000372363636363636
$cbr_(0) set random_ 0
# $cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $sink_(0)
$ns_ at 1.0 "$cbr_(0) start"
```

While this would have been suitable for this first test, as the number of nodes increased, manually editing traffic pattern files would become somewhat laborious and so it was decided to code the traffic patterns directly in the simulation Tcl scripts. As such, *drtest-2-1-cbr-11*, was not used in this simulation. Instead, `node_(0)` would be used for all sinks and the CBR generators would start at a time equal to their node number, as shown below.

```
# Load the traffic model
# source $val(cp)
# Set up the traffic flows
for {set i 1} {$i < $val(nn) } {incr i} {
    set udp_($i) [new Agent/UDP]
    $ns_ attach-agent $node_($i) $udp_($i)
    set sink_($i) [new Agent/LossMonitor]
    $ns_ attach-agent $node_(0) $sink_($i)
    set cbr_($i) [new Application/Traffic/CBR]
    $cbr_($i) set packetSize_ 512
    $cbr_($i) set interval_ [expr 0.004096/$val(bps)]
    $cbr_($i) set random_ 0
    $cbr_($i) attach-agent $udp_($i)
    $ns_ connect $udp_($i) $sink_($i)
    $ns_ at $i "$cbr_($i) start"
}
```

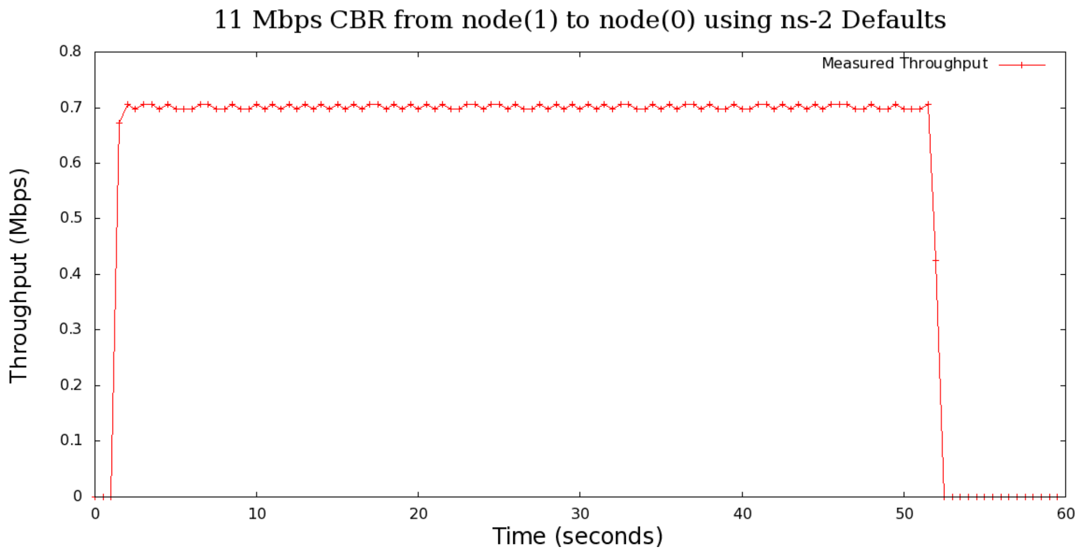


Figure 6.1: Default *ns-2* configuration — 11 Mbps node_(1) to node_(0)

Figure 6.1 shows that the WLAN is only achieving an average throughput of 702 kbps or 6.38% of the application's nominal 11 Mbps rate. This trace was produced by recording tuples of the simulation time and the number of bits recieved at the sink since the previous interval, for every 0.5 seconds of simulation time into the trace file *drtest1-node1.tr* whose file descriptor is held in `f1`, as shown here.

```
# Record data for our extra trace files
proc record {} {
    global sink_ f1
    #Get an instance of the simulator
    set ns_ [Simulator instance]
    #Set the time after which the procedure should be called again
    set time 0.5
    #How many bytes have been received by the traffic sinks?
    set bw1 [$sink_(1) set bytes_]
    #Get the current time
    set now [$ns_ now]
    #Calculate the bandwidth (in bit/s) and write it to the files
    puts $f1 "$now [expr $bw1/$time*8]"
    #Reset the bytes_ values on the traffic sinks
    $sink_(1) set bytes_ 0
    #Re-schedule the procedure
    $ns_ at [expr $now+$time] "record"
}
$ns_ at 0.0 "record"
```

6.4.1 Varying the MSDU Rate

Figure 6.2 shows the results of varying the CBR generator output from 0 to 1.2 Mbps, increasing by 10 kbps three times a second, starting the CBR generator at time 1.0 at 30 kbps. This was achieved by changing the CBR `interval_` every one third of a second, using the following code.

```
for {set i 4} {$i < $val(bps)*11} {incr i} {
    # Ramp up the bit rate so we can see what's going on
    set change [expr $i/3.0]
    set rate [expr $i/100.0]
    puts $f0 "$change, $rate"
    $ns_ at [expr $i/3.0] "$cbr_(1) set interval_ [expr 0.4096/$i]"
}
```

Figure 6.2 shows a linear increase in throughput, in proportion to the increasing rate of the CBR generator, until the performance ceiling is reached, at an

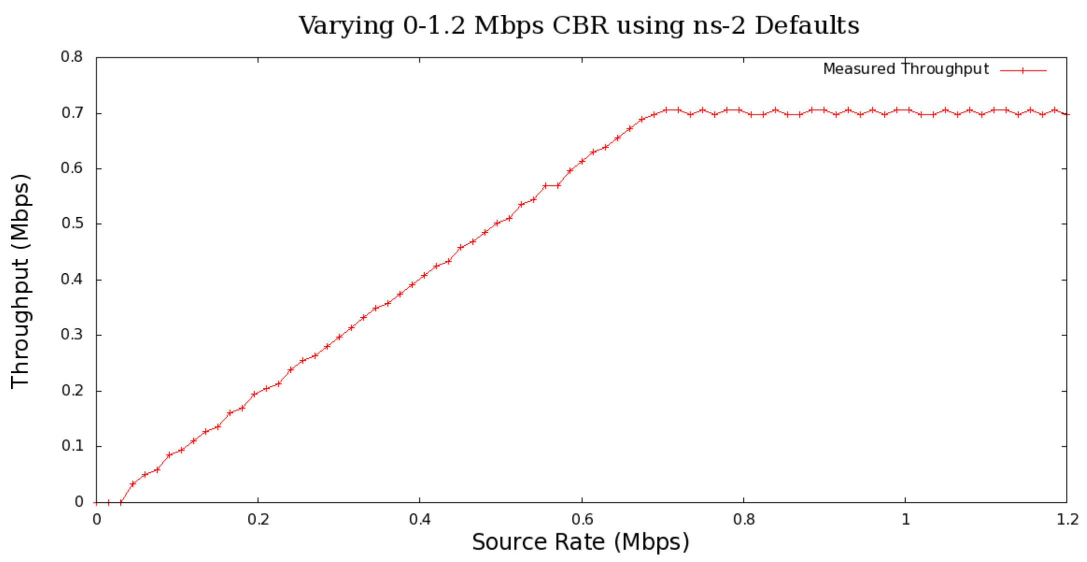


Figure 6.2: Default *ns-2* configuration — Varying 0–1.2 Mbps Source

average 702 kbps. These results indicate the throughput is not being limited by the application layer performance and other factors must be involved.

6.4.2 Default Data Rates in ns-2

The Basic Rate used for the PLCP preamble of all IEEE 802.11 frames is 1Mbps. The applicable Data Rate is used for the body of the frame, including the entire the PSDU. *ns-2*, by default, has the Data Rate for the MAC set at 2 Mbps, so to get 11 Mbps, we need:

```
Mac/802_11 set dataRate_ 11Mb
```

This was configured for *drtest3.tcl*, along with a CBR Packet Size of 1 KiB, to use:

```
$cbr_($i) set packetSize_ 1024
$cbr_($i) set interval_ [expr 0.08192/3]
```

and now increasing the CBR generator by *100* kbps three times a second:

```
for {set i 4} {$i < $val(bps)*11} {incr i} {
    # Ramp up the bit rate so we can see what's going on
    set change [expr $i/3.0]
    set rate [expr $i/10.0]
    puts $f0 "$change, $rate"
    $ns_ at [expr $i/3.0] "$cbr_(1) set interval_ [expr 0.08192/$i]"
}
```

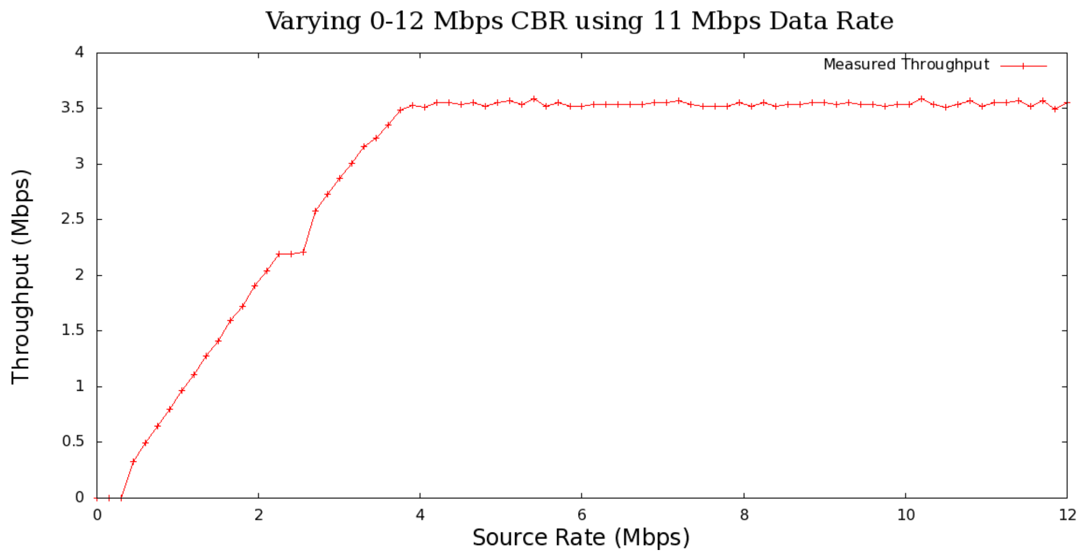


Figure 6.3: Data Rate 11 Mbps — Varying 0–12 Mbps Source

Figure 6.3 shows a five-fold improvement. We are now achieving an average 3.545 Mbps or 32.23% of our nominal 11 Mbps. Although currently simulating an IEEE 802.11b network, it was decided to test the effects of setting the Data Rate to 54 Mbps, as in an IEEE 802.11g DSSS-ODFM network. This was simulated in *drtest4.tcl*, with the only change being:

```
Mac/802_11 set dataRate_ 54Mb
```

This resulted in a further improvement to the maximum throughput, but a much lower percentage of the nominal data rate. Note that although the application bit rate was still only 12 Mbps at this stage, this has no effect on these results, as the throughput has not reached anywhere near this level.

Figure 6.4 shows a further 36% improvement to the maximum throughput to an average 4.821 Mbps. However, we are now only achieving 8.93% of our nominal 54 Mbps. Clearly, while increasing the data rate does improve throughput, the efficiency is decreasing rapidly.

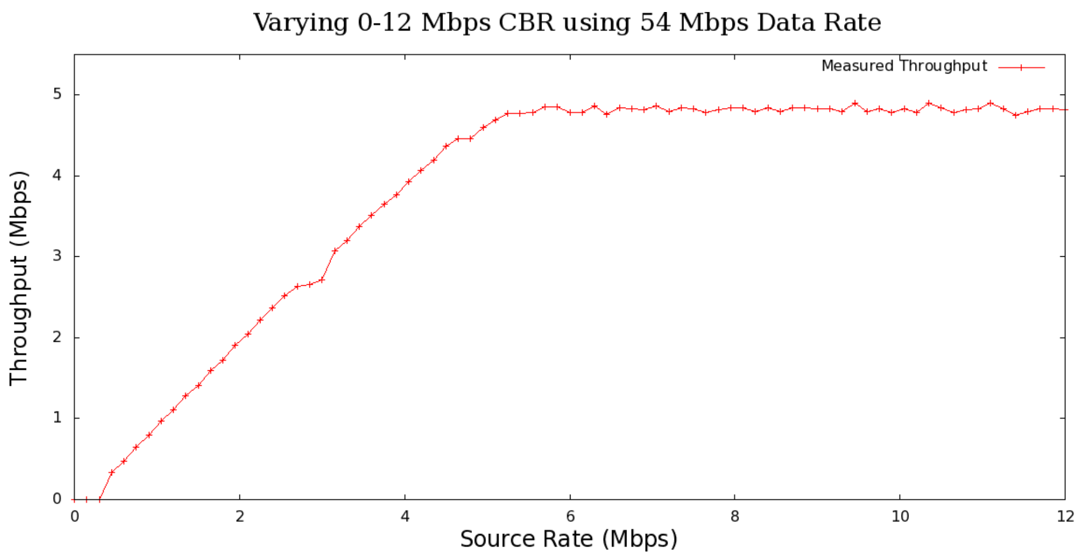


Figure 6.4: Data Rate 54 Mbps — Varying 0–12 Mbps Source

6.4.3 The RTS/CTS Threshold

The CMU/Rice *Wireless and Mobility Extensions to ns-2* [25], by default, has the RTS/CTS threshold set zero, using the RTS/CTS exchange for all data packets. This can be turned off by setting RTSThreshold_ larger than the maximum packet size. As the default fragmentation size in *ns-2* is 2346 bytes, an RTSThreshold_ of 3000 bytes will effectively turn off the RTS/CTS exchange for all packets. This was configured for *drtest5.tcl*, with the single change:

```
Mac/802_11 set RTSThreshold_ 3000
```

Figure 6.5 shows a considerable 67% improvement in throughput averaging 8.076 Mbps with no RTS/CTS exchange over Figure 6.4 with the RTS/CTS exchange for these 1 KiB data packets. This is now 14.96% of the 54 Mbps channel bandwidth.

6.4.4 The Frame Preamble

The CMU/Rice `Mac/802_11` enhancements [25], use the 144-bit DSSS frame preamble, with a 128-bit Sync field and the 16-bit SFD, by default. However the IEEE 802.11b HR/DSSS offers an optional 72-bit HR/DSSS/Short frame preamble, with a 56-bit Sync field and the 16-bit SFD; and in the IEEE 802.11g Ex-

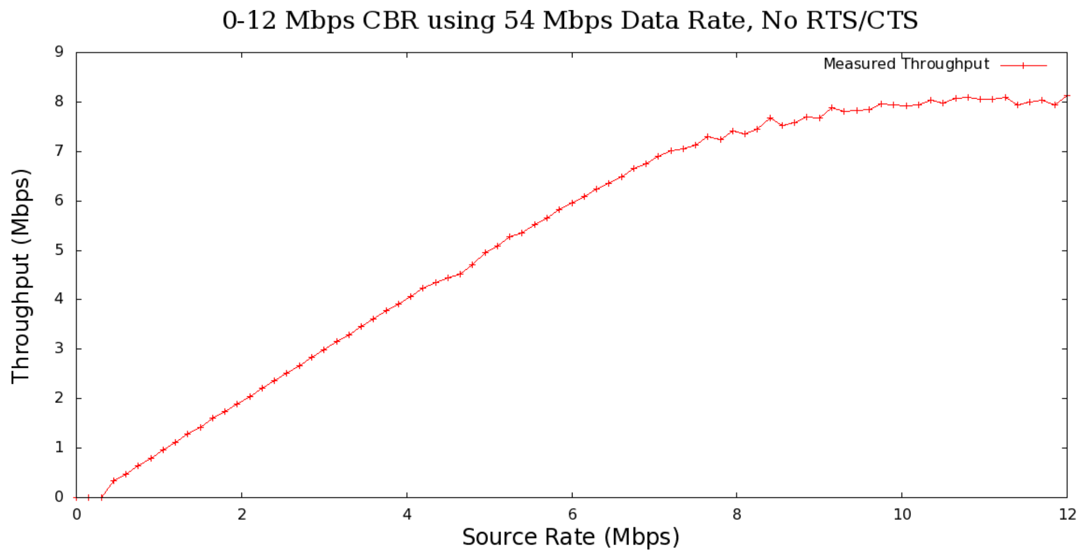


Figure 6.5: Data Rate 54 Mbps, RTS/CTS Disabled — Varying Source

tended Rate PHY (ERP) this capability is mandatory. The 72-bit short preamble was configured for *drtest6.tcl*, with:

```
Mac/802_11 set PreambleLength_ 72
```

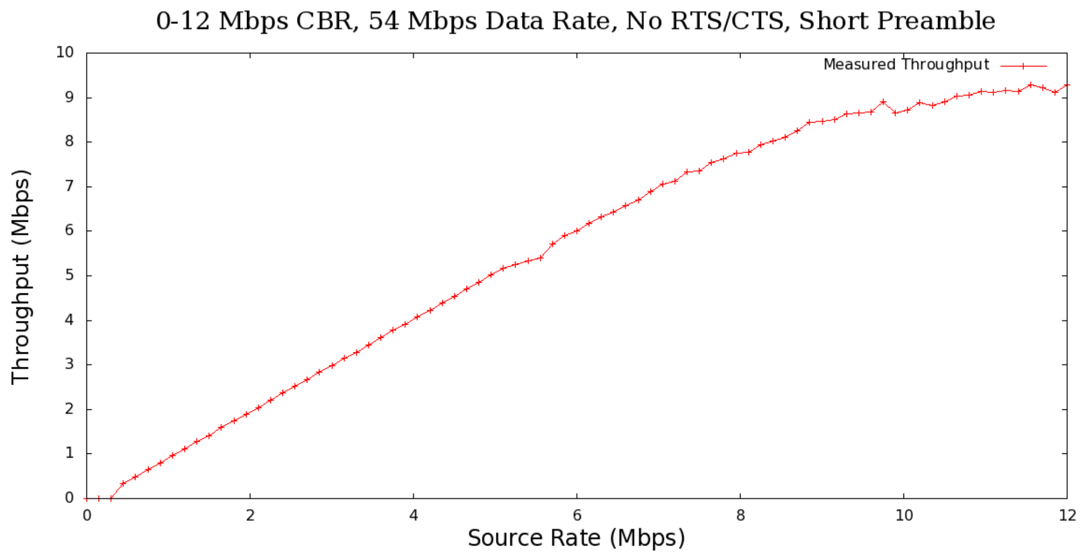


Figure 6.6: Rate 54 Mbps, No RTS/CTS, Short Preamble, Varying Source

Figure 6.6 shows a maximum throughput averaging 9.258 Mbps, a further 14.6% improvement in throughput with the short preamble over Figure 6.5 with the long preamble, for our 1 KiB payloads. This is now averaging 17.14% of the 54 Mbps channel bandwidth.

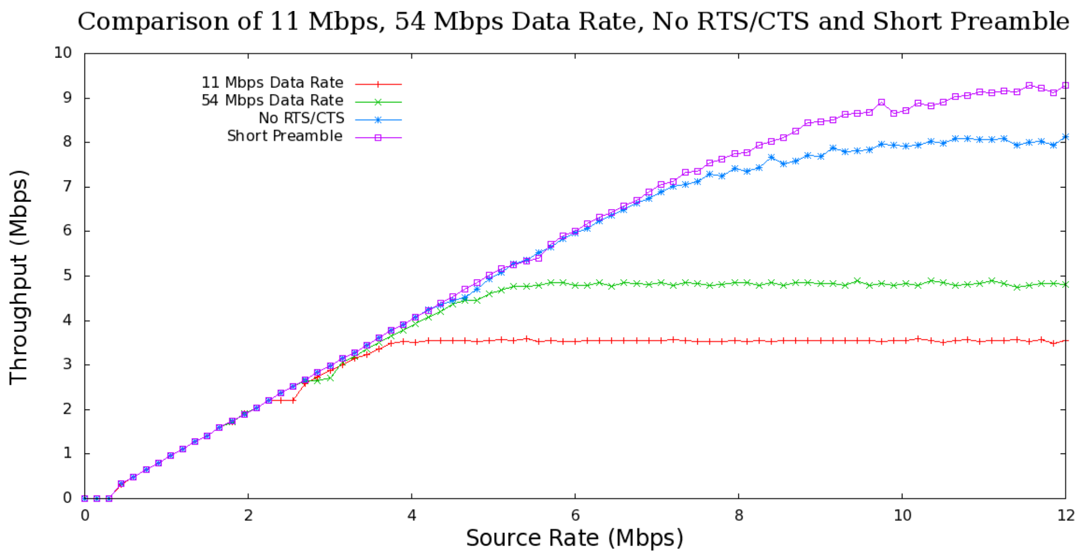


Figure 6.7: Compare 11 & 54 Mbps Data Rate, no RTS/CTS, Short Preambles

Figure 6.7 shows the cumulative effect of the changes to the frame format and transmission mode.

6.4.5 The Packet Size

Having extracted as much as we can from the frame format and transmission, we now turn our attention the size of the payload. *drtest7.tcl* was configured to vary the CBR generator payload from 50 bytes to 5000 bytes, increasing by 50 bytes two times a second, starting at time 0.0, all while maintaining a CBR of 54 Mbps. This was achieved by changing the CBR `packetSize_` every one half of a second and adjusting `interval_` to maintain 54 Mbps, viz:

```
for {set i 1} {$i < 100} {incr i} {
 # Ramp up the packet sizes so we can see what's going on
 set change [expr $i/2.0]
 set pktsze [expr ($i+1)*50]
 puts $f0 "$change, $pktsze"
 $ns_ at [expr $i/2.0] "$cbr_(1) set packetSize_ [expr ($i+1)*50]"
 $ns_ at [expr $i/2.0] "$cbr_(1) set interval_ [expr ($i+1)*0.0004/54]"
}
```

The domain of the results was converted from time to the size of the packets for the previous half second with:

```
#Calculate the size and bandwidth (in bit/s) and write it to the files
puts $f1 "[expr $now*100] [expr $bw1/$time*8]"
```

Note the terms `$now*100` recording results, compared with time `$i/2.0` and size `($i+1)*50` sending the packets. The extra one in the `($i+1)` term is required because the `proc record {}` reads the number of bytes from the *previous* half-second *up to* `$now` — so the sender starts at 50, while the receiver starts at zero and both increment by 50 every half second.

In addition, from this point on, the scene movement file used, *drtest-2-1-scen*, was modified to leave the two nodes at 1 m range for the entire simulation, eliminating all movement and remaining within the 250 m radio range throughout the simulation. The relevant settings in the scene movement file were:

```
$node_(0) set X_ 25.000000000000
$node_(0) set Y_ 175.000000000000
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 26.000000000000
$node_(1) set Y_ 175.000000000000
$node_(1) set Z_ 0.000000000000
# $ns_ at 2.000000000000 "$node_(1) setdest 325.000000000000 175.000000
000000 5.000000000000"
```

Figure 6.8 shows a near-linear increase in throughput, almost in direct proportion to the increasing size of the CBR packets, until at 1 kB a ceiling is reached, averaging 9.398 Mbps. These results indicate a 1 kB hard limit for the packet size, irrespective of any other conditions. We also note, that 1 kB (1000-byte) packets provide 9.394 Mbps throughput, while 1050-byte packets provide only 9.351 Mbps. This correlates to our previous 1 KiB (1024-byte) packets providing an average 9.258 Mbps with at most only 9.390 Mbps during the simulation. This decrease after 1 kB also indicates that some sort of fragmentation is occurring.

To verify these results *drtest8.tcl* was configured to vary the CBR generator payload from 50 bytes to *20,000* bytes, increasing by 50 bytes four times a second, starting at time 0.0 for 100 seconds, all while maintaining a CBR of 54 Mbps.

Once again, Figure 6.9 demonstrates the near-linear increase in throughput, rising with the increasing size of the CBR packets until the 1 kB ceiling is reached, and averaging 9.395 Mbps thereafter.

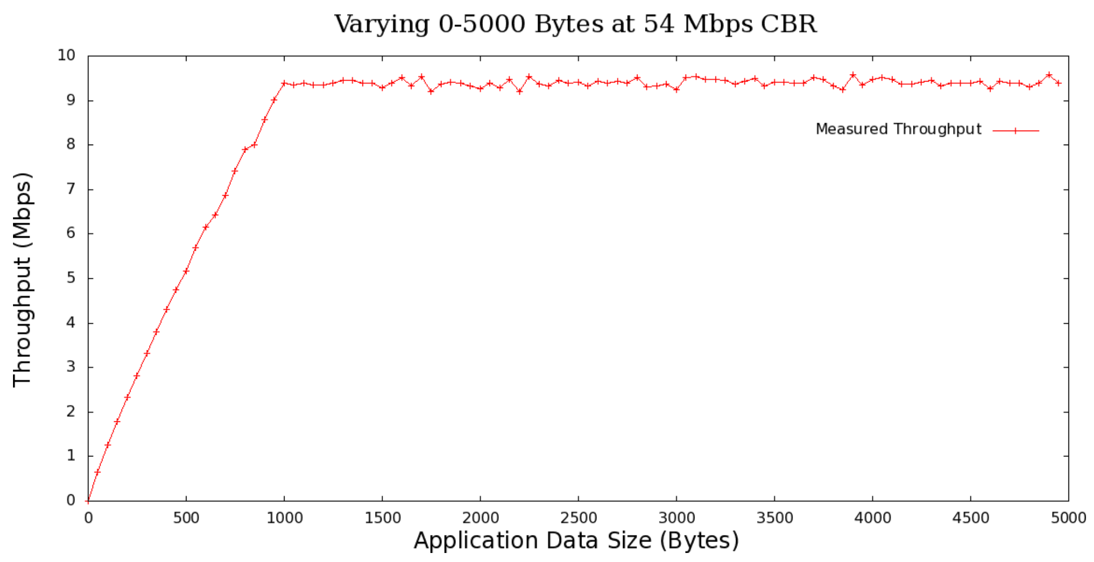


Figure 6.8: Varying Packet Size 0–5,000 Bytes

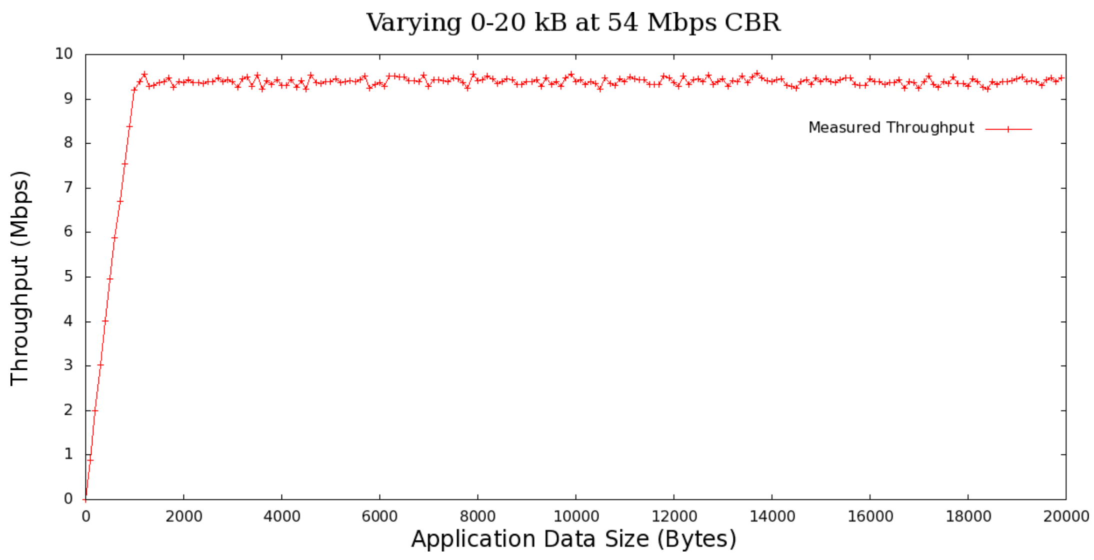


Figure 6.9: Varying Packet Size 0–20,000 Bytes

6.4.6 Fragmentation at every Layer

The defaults for most *ns-2* parameters can be found in *~ns2/tcl/lib/ns-default.tcl*, including:

```
Application/Traffic/CBR set rate_ 448Kb ;# a packet interval of 3.75ms
Application/Traffic/CBR set packetSize_ 210
Application/Traffic/CBR set random_ 0
Application/Traffic/CBR set maxpkts_ 268435456; # 0x10000000

Agent/UDP set packetSize_ 1000

Agent/TCP set packetSize_ 1000
Agent/TCP set tcpip_base_hdr_size_ 40
Agent/TCP set ts_option_size_ 10

LL set mindelay_                 50us
LL set delay_                    25us
LL set bandwidth_                0       ;# not used

Mac/802_11 set PreambleLength_   144       ;# 144 bit
Mac/802_11 set PLCPHeaderLength_ 48        ;# 48 bits
Mac/802_11 set PLCPDataRate_     1.0e6     ;# 1Mbps
Mac/802_11 set RTSThreshold_     0         ;# bytes

Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set bandwidth_ 2e6
Phy/WirelessPhy set Pt_ 0.28183815
```

Note that all “sizes” (`packetSize_`, `tcpip_base_hdr_size_`, `ts_option_size_`) are given in bytes, so all interval/rate calculations require these to be multiplied by eight, viz. $rate\_ = \frac{8*size\_}{interval\_}$ and $interval\_ = \frac{8*size\_}{rate\_}$; whereas the “lengths” (`PreambleLength_`, `PLCPHeaderLength_`) are given in bits. This is not necessarily universal, just in the parameters we are dealing with here.

Note that although *ns-2* maintains numerous headers for each packet, the various traffic sources, transport agents and the link layer do not add the UDP/TCP, IP or LLC header lengths, $\binom{8}{20}+20+8 = \binom{36}{48}$ bytes, on to the simulated packet’s `size_` field, so the `packetSize_` parameters are effectively the size of the MSDU passed to the MAC layer.

The `random_` parameter in the CBR generator pseudo-randomly changes the interval by up to half its value for each packet. This is not desired in these trials and is kept off.

The `mindelay_` parameter in the link layer, `LL`, was introduced by the *CMU Monarch Project's Wireless and Mobility Extensions to ns* for the "minimum time the `LL` object will hold onto a packet" [25], but is now obsolete and although set to the value 50 µs in 49 separate locations in the *ns-2.33* sources, is not then referenced in any source in the *ns-2.33* distribution. This function is performed by the inherited `delay_` parameter from the parent class.

`LL` simulates the link layer and is a subclass of `LinkDelay`. The `LinkDelay` class is derived from the `Connector` class and simulates a link with latency. `LinkDelay` defines `bandwidth_`, the bandwidth of underlying link in bits per second and `delay_`, the line latency for the link. `LL` inherits these, but being part of the protocol stack, as opposed to a physical link, does not use the inherited `bandwidth_`, which is set by default to zero. The inherited `delay_` simulates the overhead of the link layer.

The wireless PHY has the transmitted signal power, `Pt_`, a minimum carrier sense threshold, `CSThresh_`, and a minimum receiving power threshold, `RXThresh_`, all in watts, and requires a signal to have received strength at least the capture threshold ratio, `CPThresh_`, greater than any other interfering signal in order to succeed. Note that `CPThresh_` is a direct ratio — not a logarithmic ratio in decibels (dB) as given in *~ns2/mac/wireless-phy.h*, nor a value in watts as given in [164].

Like the `LL`, the `WirelessPhy` does not use the `bandwidth_` parameter at all, even though it is set in defaults to 2 Mbps. The current (*ns-2.33*) code for the `WirelessPhy` has the binding of the *Tcl* and *C++* `bandwidth_` parameters commented-out. The only place in the current wireless *ns-2* modules where the `bandwidth_` parameter is used, is at the MAC layer, where it is used to determine transmit time, by the parent `Mac` class in *~ns2/mac/mac.h* and as a fallback for the `basicRate_` and `dataRate_` by the `Mac/802_11` class in *~ns2/mac/mac-802_11.cc*, if they have not already been set.

Other relevant parameters can be found in *~ns2/mac/mac-802_11.h*, including:

```
#define DSSS_MaxPropagationDelay        0.000002        // 2us
#define MAC_FragmentationThreshold      2346            // bytes
```

From all of these settings, we see that although the `Mac/802_11` default fragmentation size is 2346 bytes and we are manually setting the CBR packet size as desired, the default UDP payload size is 1000 bytes. Fragmentation at the Transport Layer is now limiting us. This was removed by configuring *drtest9.tcl* with:

```
Agent/UDP set packetSize_ 20000
```

and to keep the RTS/CTS exchange turned off for these large test packets,

```
Mac/802_11 set RTSThreshold_ 20000
```

was also included.

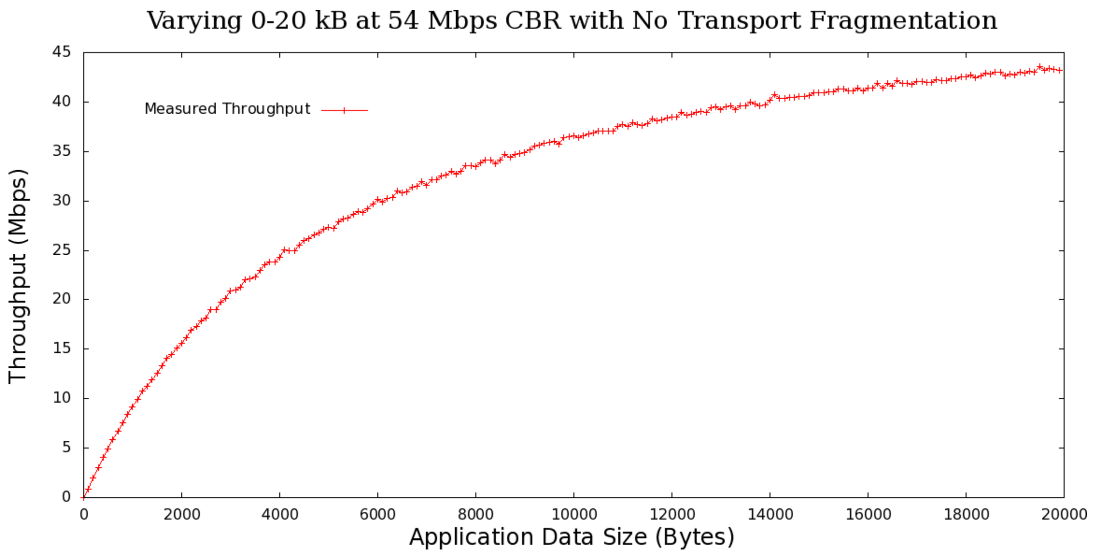


Figure 6.10: Varying Packet Size with No Transport Layer Fragmentation

Figure 6.10 now shows asymptotic gain with increasing packet size approaching 44 Mbps as packet size approaches 20 kB, over 80% of the nominal channel bandwidth. The actual asymptote, as packet size approaches infinity, can not be determined from these data, but should be near 54 Mbps for an infinitely large packet with only a single set of headers and channel negotiation. This is clearly the bulk of the efficiencies to be gained by manipulating packets and transmission characteristics. The remaining losses are due to necessary encapsulation data, radio synchronisation overheads, the various interframe spaces and collision avoidance protocols.

6.5 Establishing the Baseline

A series of simulations were undertaken to establish the status quo for the current conditions as a baseline set of control values to compare against the amended protocols.

6.5.1 The Maximum MSDU Size

The maximum IEEE 802.11 [9, p. 30] MSDU size is 2304 octets[14]. It is the maximum size of the MAC Service Data Unit, i.e. the PDU from the LLC, before any MAC-layer encryption is applied[15]. The IEEE 802.11b/g maximum MSDU size is the same. The maximum IEEE 802.11a [71, p. 5] MPDU size is 4095 octets. In this case it is the MAC Protocol Data Unit, i.e. the PSDU, not the MSDU — but still, considerably larger. IEEE 802.11n allows MPDU aggregation and so allows very large payloads [77].

As 2304 octets is the largest MSDU that can be accommodated by all of the protocols, it was selected for the maximum transport layer PDU. The next simulation, *drtesta.tcl*, was configured with:

```
Agent/UDP set packetSize_ 2304
```

which, as discussed above, actually simulates a 2304 octet maximum MSDU (LLC PDU) size. This was configured in conjunction with:

```
Mac/802_11 set RTSThreshold_ 3000
```

to support this with no RTS/CTS exchanges. Figure 6.11 shows the results of the same CBR generator payload varying from 50 bytes to 20,000 bytes, increasing by 50 bytes four times a second, while maintaining a CBR of 54 Mbps, but limited to a maximum MSDU of 2304 bytes

Figure 6.11 shows the initial curve of Figure 6.10 as packet size increases from 50 to the maximum MSDU size of 2304 bytes. However from this point on, the throughput then levels at an average 17.52 Mbps. This achieves a throughput of 32.44% of the channel bandwith.

[14]The maximum MSDU size of 2304 octets was to allow application data of up to 2048 octets and up to 256 octets of upper-layer headers.

[15]Using WEP adds 8 octets, TKIP adds 20 octets, and CCMP adds 16 octets; rendering maximum data unit sizes of 2312, 2324, and 2320 octets respectively.

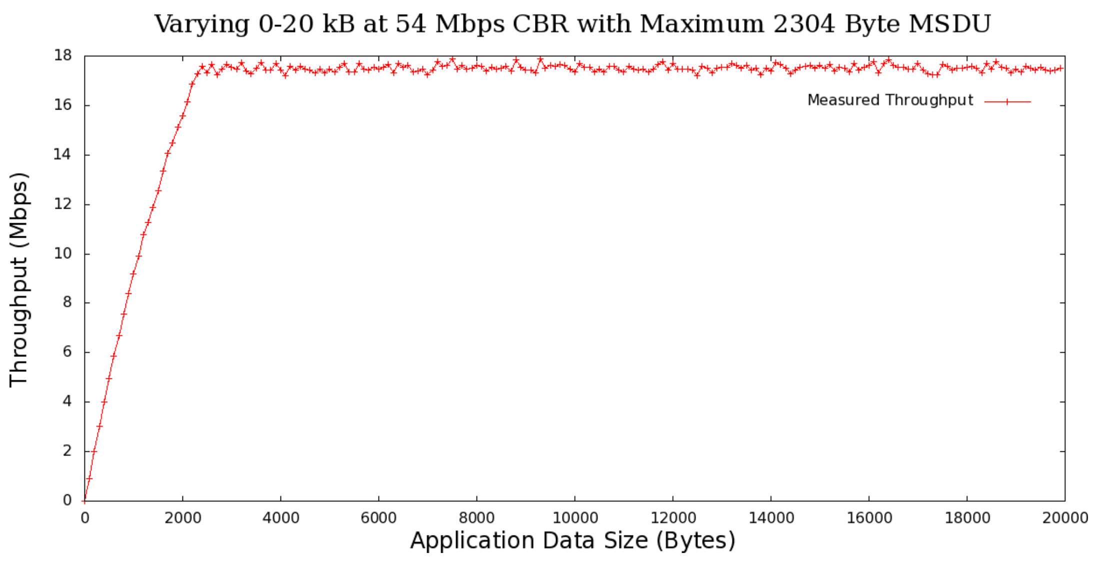


Figure 6.11: Varying Packet Size with Maximum 2304 Byte MSDU

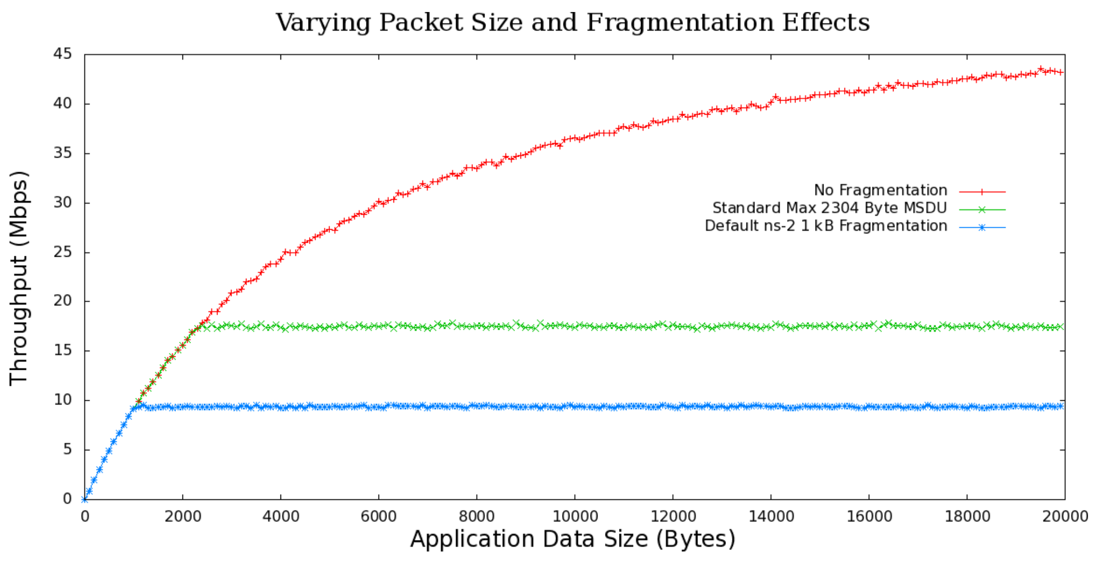


Figure 6.12: The Effects of Fragmentation

Figure 6.12 compares the fragmentation effects, showing the achieved throughput against the application data size, firstly with no fragmentation, then with the chosen control condition with a maximum 2304 byte MSDU and with the ns-2 default fragmentation at 1000 bytes.

These three traces were all established using a 54 Mbps Data Rate and show the clear truncation of effective bandwidth caused by MSDU fragmentation. The unfragmented trace demonstrates that these effects are not the result of a limitation of the chosen PHY to transfer bits at a given rate, but instead, the two fragmented traces show a limitation on the number of packets that can be squeezed into the remaining bandwidth when every packet has a fixed maximum size with a fixed overhead and acknowledgement, plus contention protocols.

6.5.2 The Effects of the Channel Data Rate

Having determined and set the maximum MSDU size, a series of simulations were then conducted to gather detailed data for packet sizes varying from 50 bytes to 2500 bytes, increasing by 50 bytes every second, generated at 54 Mbps, with PHY data rates of 1, 2, 11 and 54 Mbps.

Figure 6.13 shows the results of the simulation for a 1 Mbps wireless channel configured with *drtestb.tcl*, as follows:

```
Agent/UDP set packetSize_ 2304
LL set delay_ 25us
Mac/802_11 set dataRate_ 1Mb
Mac/802_11 set RTSThreshold_ 3000
Mac/802_11 set PreambleLength_ 72
```

Figure 6.13 again shows asymptotic gain with increasing packet size approaching 955 kbps as packet size approaches the 2304 byte maximum and then maintaining an average 954.1 kbps throughput at maximum packet size, utilising 95.41% of the nominal channel bandwith.

Figure 6.14 shows the results of the simulation for a 2 Mbps wireless channel configured with *drtestc.tcl*, using `Mac/802_11 set dataRate_ 2Mb`

Figure 6.14 shows mostly asymptotic gain with the increasing packet size, approaching an average 1.830 Mbps as packet size approaches the 2304 byte maximum and then maintains an average 1.834 Mbps throughput once the maximum packet size has been reached, utilising 91.72% of the nominal channel bandwith.

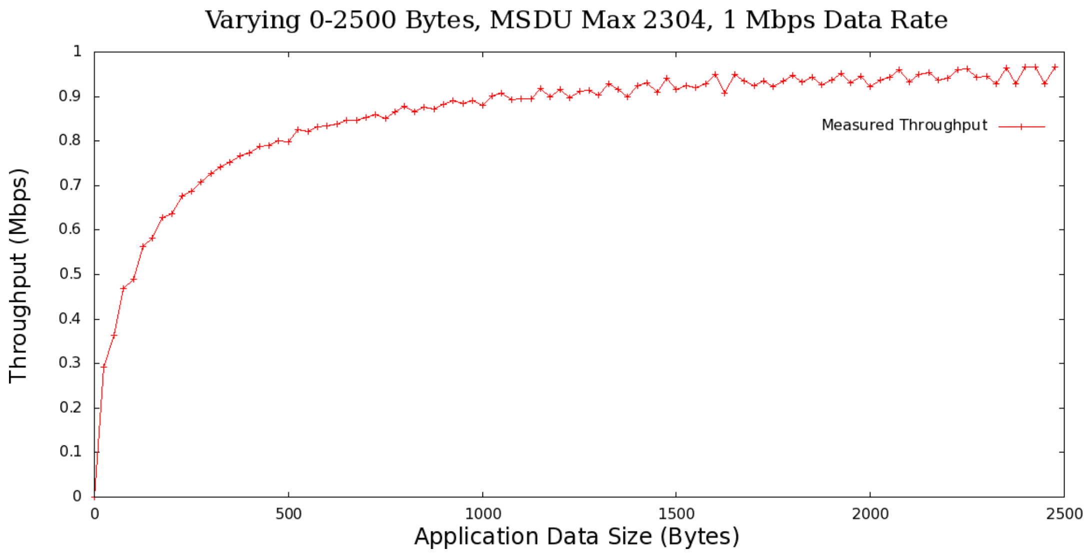


Figure 6.13: Varying MSDU Size (Max 2304) at 1 Mbps Data Rate

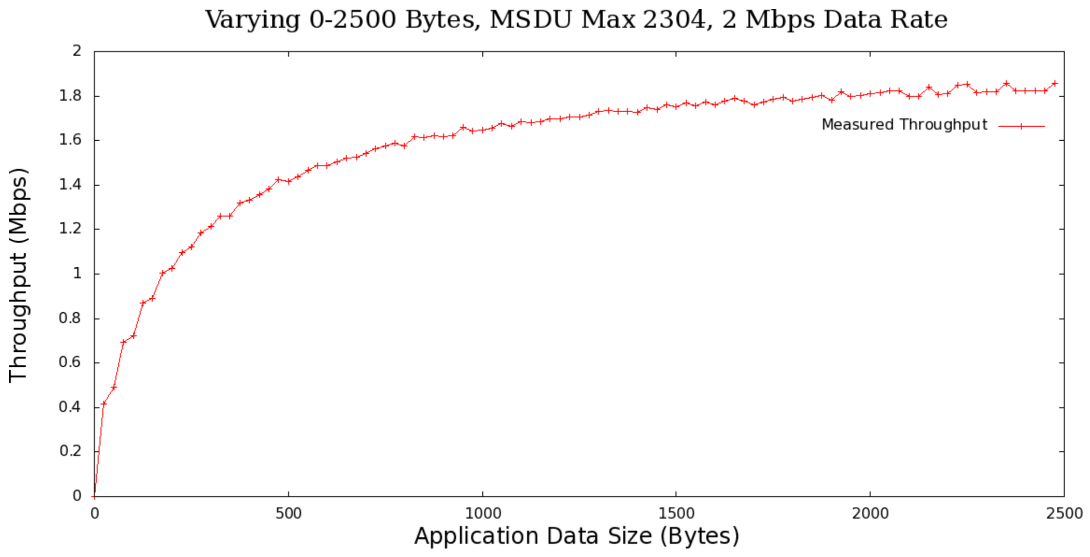


Figure 6.14: Varying MSDU Size (Max 2304) at 2 Mbps Data Rate

Figure 6.15 shows the results of the simulation for a 11 Mbps wireless channel configured with *drtestd.tcl*, using `Mac/802_11 set dataRate_ 11Mb`.

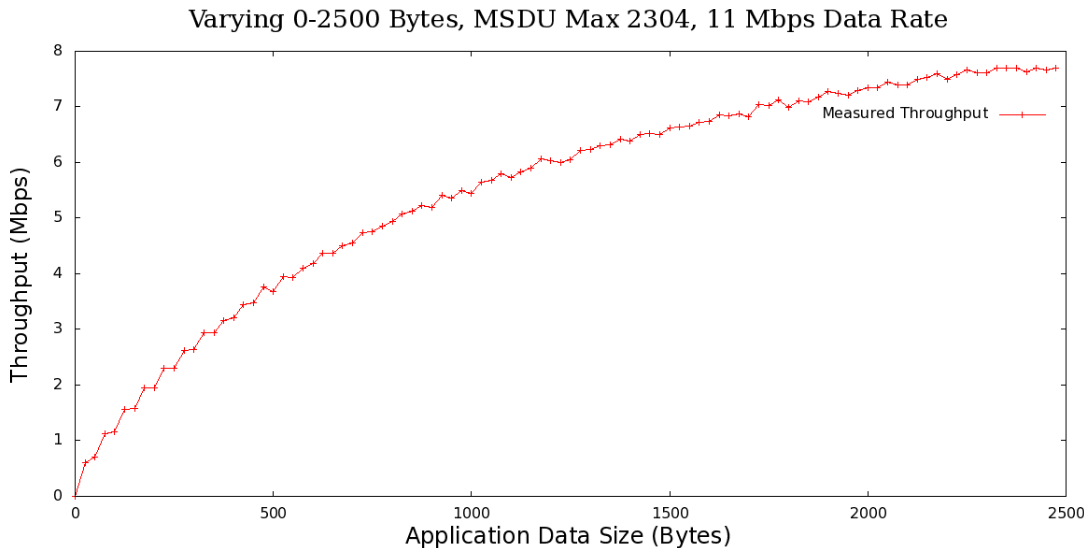


Figure 6.15: Varying MSDU Size (Max 2304) at 11 Mbps Data Rate

Figure 6.15 shows exponential limit gain as packet size approaches the 2304-byte maximum and then maintains an average 7.678 Mbps throughput thereon, utilising 69.80% of the nominal channel bandwith.

These data correlate closely with both the simulated and experimental data produced by Joshua Robinson in *Making NS-2 simulate an 802.11b link* [153].

Figure 6.16 shows *drteste.tcl* configured with a 54 Mbps wireless channel, using `Mac/802_11 set dataRate_ 54Mb`.

Figure 6.16 shows the near-linear gain with increasing packet size of the initial component of an exponential limit curve as packet size approaches the 2304-byte maximum and then maintaining an average 17.48 Mbps soon after.

As these last four simulations have ended soon after the 2304-byte maximum was reached, these terminal values have been averaged over only a half-dozen samples. The more detailed 54 Mbps channel results of Figure 6.11 give a more accurate final average (over 176 samples) of 17.52 Mbps, utilising 32.44% of the nominal channel bandwith.

Summarising these, Figure 6.17 compares the throughput obtained with MAC data rates of 1, 2, 11 and 54 Mbps on the wireless channel.

For each of the MAC data rates of 1, 2, 11 and 54 Mbps, Figure 6.17 graphs the throughput as the ordinate in Mbps on the Y-axis, against the source ap-

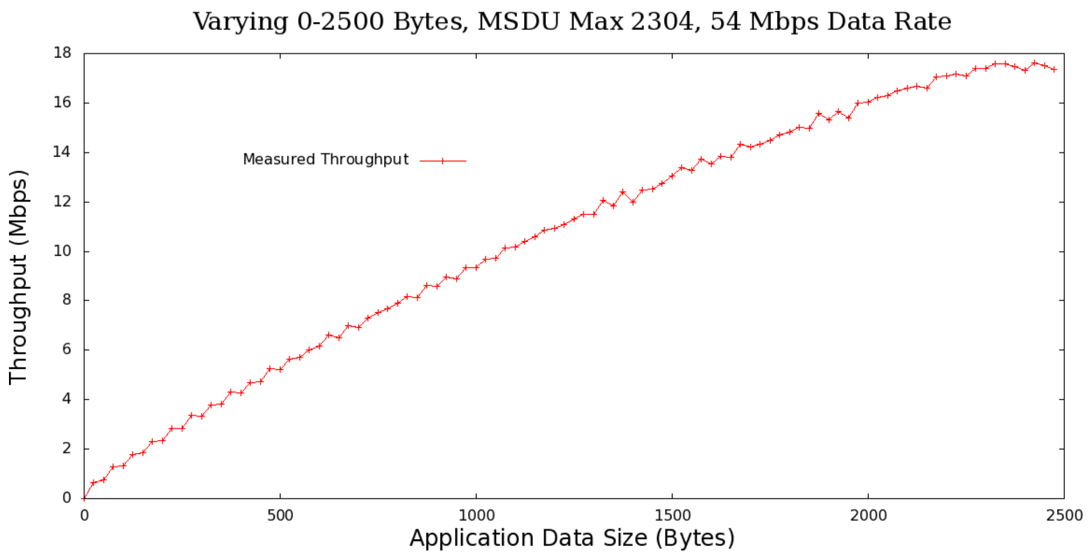


Figure 6.16: Varying MSDU Size (Max 2304) at 54 Mbps Data Rate

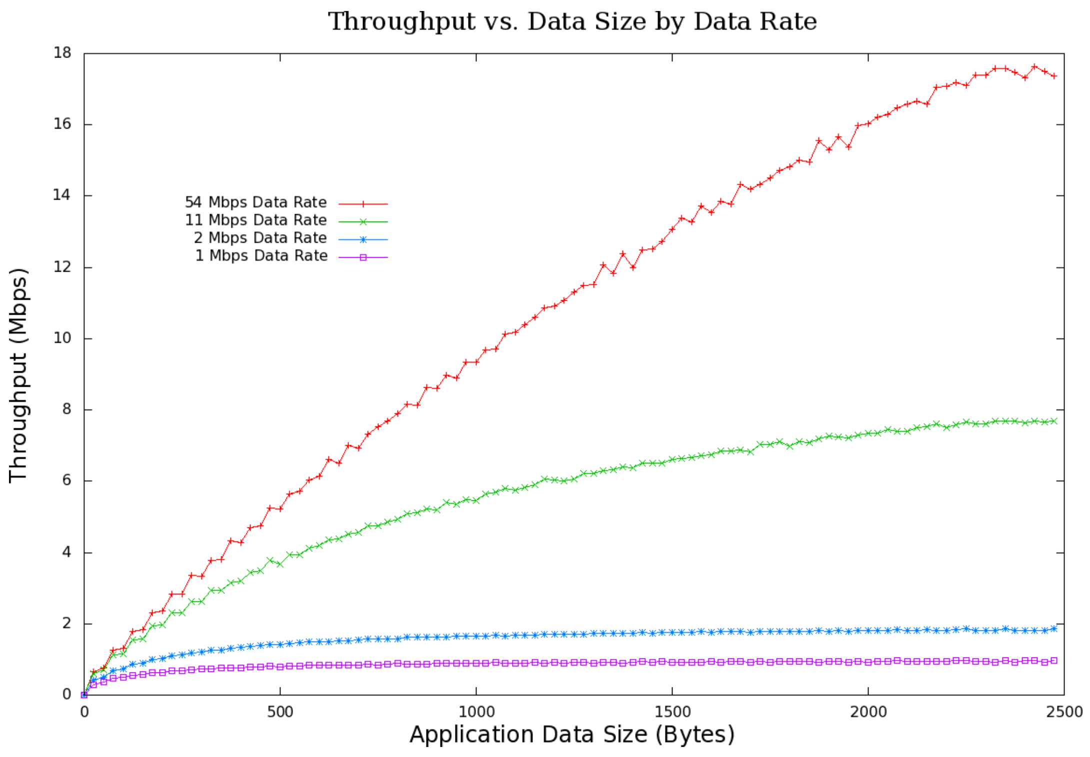


Figure 6.17: Comparative Throughput vs. Size for 1, 2, 11 and 54 Mbps Rates

plication packet size as the abscissa in kB on the X-axis. At 2.304 kB, for all MAC data rates, the packets are fragmented and the characteristic changes to an average constant level.

6.5.3 The Effects of the Application Data Rate

Next, baseline tests were performed for fixed 2048-byte (16,384-bit) packets generated at intervals varying to produce application data rates of 0 to 60 Mbps. This is achieved by actually starting from 0.5 Mbps, increasing by 0.5 Mbps every half second, to 60 Mbps, measuring at the end of each half second and recording from time 0. Figure 6.18 shows the results for *drtestf.tcl* configured with fixed STAs at 1 m range, and varying source data rates, as follows:

```
Agent/UDP set packetSize_ 2304
Mac/802_11 set dataRate_ 54Mb
Mac/802_11 set RTSThreshold_ 3000
Mac/802_11 set PreambleLength_ 72
$cbr_(1) set packetSize_ 2048
$cbr_(1) set interval_ [expr 0.016384/0.5]
for {set i 1} {$i < 120} {incr i} {
    $ns_ at [expr $i/2.0] "$cbr_(1) set interval_ [expr 0.016384/(($i+1
    )*0.5)]"
}
```

Figure 6.18 shows throughput directly proportional to CBR generation rate, up to 16 Mbps for both generation rate and throughput, but then settling to an average 16.16 Mbps throughput thereafter for the 2048-byte packets, utilising 29.93% of the nominal channel bandwith.

Note that the 1% "gain" in performance is due to the DSDV routing functions in *ns-2* adding a 20-byte IP header[16] on top of the existing segment that never gets stripped off at the receiver because the routing functions never see a received packet addressed for their own node. As such, these unstripped headers are counted in the received bytes. This means we are effectively transferring "2068-byte packets" as far as the calculations are concerned, in the same times we

[16]Both the DSDV and AODV routing protocols add a 20-byte IP header. In fact, the very first packet of the simulation gets 2 x 20-byte additions due to a queuing bug in *ns-2*'s DSDV implementation.

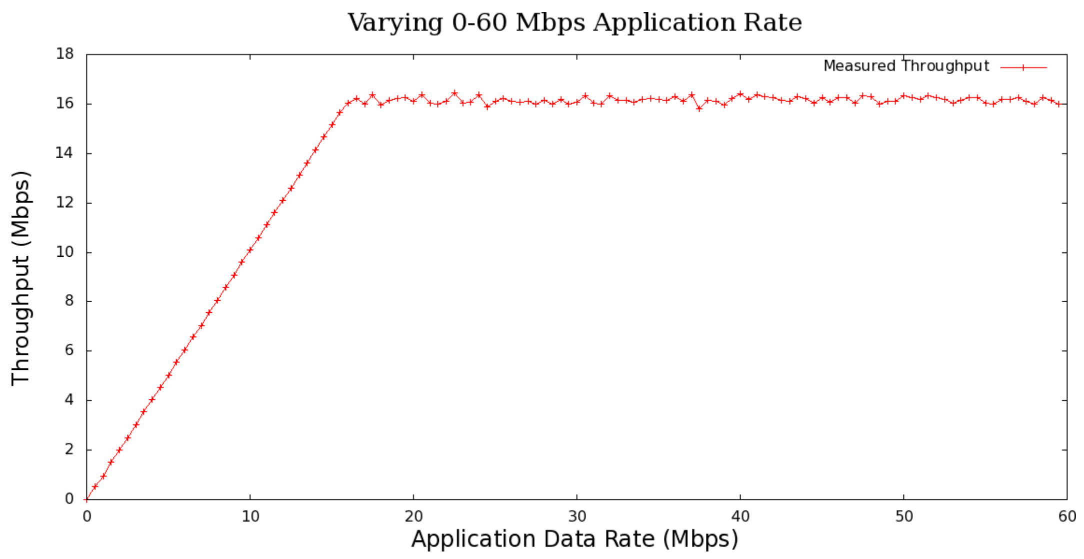


Figure 6.18: Varying Application Data Rate for Fixed STA at 1 m Range

intended to send 2048-byte packets, which gives an effective source data rate of $\frac{2068}{2048} * 16 = 16.16$ Mbps.

This behaviour is confirmed by the trace file *drtestf-out.tr*.

```
s 0.000000000 _1_ AGT  --- 0 cbr 2048 [0 0 0 0] ------- [1:0 0:0 32 0]
r 0.004255770 _0_ AGT  --- 0 cbr 2088 [f2 0 1 800] ------- [1:0 0:0 32
s 0.032768000 _1_ AGT  --- 2 cbr 2048 [0 0 0 0] ------- [1:0 0:0 32 0]
r 0.033339411 _0_ AGT  --- 2 cbr 2068 [f2 0 1 800] ------- [1:0 0:0 32
s 0.065536000 _1_ AGT  --- 3 cbr 2048 [0 0 0 0] ------- [1:0 0:0 32 0]
r 0.066227411 _0_ AGT  --- 3 cbr 2068 [f2 0 1 800] ------- [1:0 0:0 32
```

6.6 Simulating Infrastructure Mode

Note that this work does not use Ilango Purushothaman and Sumit Roy's new *ns-2.33* 802.11 infrastructure code [165–167]. The infrastructure code, while operational, is limited by the other functions of *ns-2*, such as no multi-channel or DSS capabilities, and so provides no advantage in this work.

Where desired, infrastructure mode was simulated by configuring all traffic for all nodes to pass to and from `node_(0)`, fixed at the centre of the topology, which thereby acted as the AP.

6.7 Development of the Analysis Tools

The *ns-2* trace file contains information recorded on command by options within the TCL components instantiating the wireless nodes. All simulations were performed with only `agentTrace ON`, recording the events at the UDP/TCP agents:

```
$ns_ node-config ... \
                 -agentTrace ON \
                 -routerTrace OFF \
                 -macTrace OFF \
                 -movementTrace OFF
```

However, where more detailed analysis was required, such as in the previous case, `routerTrace` and `macTrace` were added, to provide details of all packets entering and leaving the agents, routing functions and the MAC sub-layer:

```
$ns_ node-config ... \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace OFF
```

This gives a trace providing packets “s”ent, “r”eceived or “d”ropped, event time, _node_, subsystem (AGT or RTR or MAC), an error indication or `---`, event (packet) id, packet type, packet size in bytes and the MAC details [(in hex) duration field (0 µs or f2 = 242 µs SIFS + ACK), receiver address (RA), transmitter address (TA) and ether-type], followed by:

- for ARP packets, `-------`, [“REQUEST” or “REPLY”, (in decimal) ARP source address:source port and ARP target address:target port];
- for IP packets, `-------`, [(in decimal) IP source address:source port, IP destination address:destination port, IP time-to-live (32) and next hop];
- for IP route advertisements, `-------`, [(in decimal) IP source address:source port (255), IP destination address (-1):destination port (255), IP time-to-live (32) and next hop (0)];

- for TCP packets, `-------`, [(in decimal) IP source address:source port, IP destination address:destination port, IP time-to-live (32), next hop or 0], [Sequence number, Ack number], number of forwards and (mobile) optimal number of forwards[17];

This results in output of the form,

```
s 0.089076526 _3_ RTR  --- 0 message 32 [0 0 0 0] ------- [3:255 -1:255 32 0]
s 0.089171526 _3_ MAC  --- 0 message 81 [0 ffffffff 3 800] ------- [3:255 -1:255 32 0]
r 0.089819879 _1_ MAC  --- 0 message 32 [0 ffffffff 3 800] ------- [3:255 -1:255 32 0]
r 0.089819999 _6_ MAC  --- 0 message 32 [0 ffffffff 3 800] ------- [3:255 -1:255 32 0]
r 0.089844879 _1_ RTR  --- 0 message 32 [0 ffffffff 3 800] ------- [3:255 -1:255 32 0]
r 0.089844999 _6_ RTR  --- 0 message 32 [0 ffffffff 3 800] ------- [3:255 -1:255 32 0]
s 5.000000000 _1_ AGT  --- 11 tcp 40 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
r 5.000000000 _1_ RTR  --- 11 tcp 40 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
s 5.000000000 _1_ RTR  --- 11 tcp 60 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
s 5.000175000 _1_ MAC  --- 0 ARP 77 [0 ffffffff 1 806] ------- [REQUEST 1/1 0/0]
r 5.000791451 _0_ MAC  --- 0 ARP 28 [0 ffffffff 1 806] ------- [REQUEST 1/1 0/0]
r 5.000791690 _6_ MAC  --- 0 ARP 28 [0 ffffffff 1 806] ------- [REQUEST 1/1 0/0]
s 5.000886451 _0_ MAC  --- 0 ARP 77 [f2 1 0 806] ------- [REPLY 0/0 1/1]
r 5.001016088 _1_ MAC  --- 0 ARP 28 [f2 1 0 806] ------- [REPLY 0/0 1/1]
s 5.001026088 _1_ MAC  --- 0 ACK 29 [0 0 0 0]
r 5.001258539 _0_ MAC  --- 0 ACK 29 [0 0 0 0]
s 5.001568088 _1_ MAC  --- 11 tcp 109 [f2 0 1 800] ------- [1:0 0:0 32 0] [0 0] 0 0
r 5.001702465 _0_ MAC  --- 11 tcp 60 [f2 0 1 800] ------- [1:0 0:0 32 0] [0 0] 1 0
s 5.001712465 _0_ MAC  --- 0 ACK 29 [0 1 0 0]
r 5.001727465 _0_ AGT  --- 11 tcp 60 [f2 0 1 800] ------- [1:0 0:0 32 0] [0 0] 1 0
s 5.001727465 _0_ AGT  --- 12 ack 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 5.001727465 _0_ RTR  --- 12 ack 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 5.001727465 _0_ RTR  --- 12 ack 60 [0 0 0 0] ------- [0:0 1:0 32 1] [0 0] 0 0
r 5.001944916 _1_ MAC  --- 0 ACK 29 [0 1 0 0]
s 5.002214465 _0_ MAC  --- 12 ack 109 [f2 1 0 800] ------- [0:0 1:0 32 1] [0 0] 0 0
r 5.002348842 _1_ MAC  --- 12 ack 60 [f2 1 0 800] ------- [0:0 1:0 32 1] [0 0] 1 0
s 5.002358842 _1_ MAC  --- 0 ACK 29 [0 0 0 0]
r 5.002373842 _1_ AGT  --- 12 ack 60 [f2 1 0 800] ------- [0:0 1:0 32 1] [0 0] 1 0
s 5.002373842 _1_ AGT  --- 13 tcp 552 [0 0 0 0] ------- [1:0 0:0 32 0] [1 0] 0 0
r 5.002373842 _1_ RTR  --- 13 tcp 552 [0 0 0 0] ------- [1:0 0:0 32 0] [1 0] 0 0
s 5.002373842 _1_ RTR  --- 13 tcp 572 [0 0 0 0] ------- [1:0 0:0 32 0] [1 0] 0 0
s 5.002373842 _1_ AGT  --- 14 tcp 552 [0 0 0 0] ------- [1:0 0:0 32 0] [2 0] 0 0
r 5.002373842 _1_ RTR  --- 14 tcp 552 [0 0 0 0] ------- [1:0 0:0 32 0] [2 0] 0 0
s 5.002373842 _1_ RTR  --- 14 tcp 572 [0 0 0 0] ------- [1:0 0:0 32 0] [2 0] 0 0
r 5.002591294 _0_ MAC  --- 0 ACK 29 [0 0 0 0]
s 5.002840842 _1_ MAC  --- 13 tcp 621 [f2 0 1 800] ------- [1:0 0:0 32 0] [1 0] 0 0
r 5.003051071 _0_ MAC  --- 13 tcp 572 [f2 0 1 800] ------- [1:0 0:0 32 0] [1 0] 1 0
s 5.003061071 _0_ MAC  --- 0 ACK 29 [0 1 0 0]
r 5.003076071 _0_ AGT  --- 13 tcp 572 [f2 0 1 800] ------- [1:0 0:0 32 0] [1 0] 1 0
s 5.003076071 _0_ AGT  --- 15 ack 40 [0 0 0 0] ------- [0:0 1:0 32 0] [1 0] 0 0
r 5.003076071 _0_ RTR  --- 15 ack 40 [0 0 0 0] ------- [0:0 1:0 32 0] [1 0] 0 0
s 5.003076071 _0_ RTR  --- 15 ack 60 [0 0 0 0] ------- [0:0 1:0 32 1] [1 0] 0 0
r 5.003293523 _1_ MAC  --- 0 ACK 29 [0 1 0 0]
```

[17]The optimal number of forwards value is the minimum number of hops calculated by the mobile General Operations Director in *~ns2/mobile/god.cc*, but is never used.

These trace files are typically analysed using data manipulation tools, such as *grep*, *sed* and *awk*[18]. These scripts are usually employed to extract and aggregate or analyse a particular piece of information, such as throughput for a particular node or the channel itself or latency or number of lost/delayed/retransmitted packets and so forth.

In this case, the author needed a tool to assist in visualising the complete picture; to determine which individual packets were being delayed and why; to display what interactions were occurring with multiple nodes contending for the channel; or how the choices of contention backoff affected channel acquisition and the like. The author needed the chronological layout of the trace output itself, but with additional information, such as time elapsed since the previous event, packet transmission time, propagation time and the number of slots actually chosen by the MAC subsystem from the available contention window.

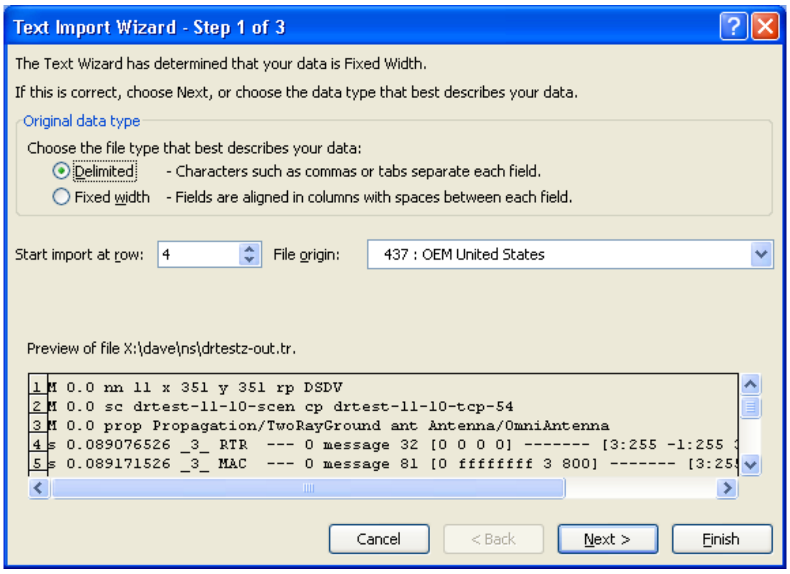
To include this information in the trace output would require considerable re-engineering of the trace functions and integrating these with the MAC subsystems. The result would be incompatible with any other version of *ns-2* and the trace output would also be incompatible with any other analysis tools currently available. Such a single-use need would unlikely be worth the effort required to develop and instantiate it.

The author determined the additional information could be extracted from the existing trace output itself and re-incorporated into that existing output to enhance the utility of this resource.

First the trace was formatted from the simple space-delimited list into a tabular layout by importing the data into a spreadsheet tool. Microsoft Excel was used, but any spreadsheet tool would serve the purpose. The data was imported with whitespace as the delimiter, as shown in Figures 6.19 and 6.20. This arranged the data fields to provide visual clarity, as shown in Figure 6.21.

To rectify any import errors, all occurrences of “1-Jan” were reverted back to the original “1/1” data. Next a new third column was inserted and all values were set with the formula $C_x = B_x - B_{x-1}$. From this elapsed time, the number of slot times for the backoff were determined for every data packet sent from a MAC subsystem by subtracting the time for the LLC delay and the DIFS and dividing the result by the slot time, all in µs, with the formula $Y_x = (C_x - 0.000075)/0.00002$, as shown in Figure 6.22.

[18] *awk* is named after its original authors, Alfred V. Aho, Peter J. Weinberger and Brian W. Kernighan.



Text Import Wizard - Step 1 of 3

The Text Wizard has determined that your data is Fixed Width.

If this is correct, choose Next, or choose the data type that best describes your data.

Original data type

Choose the file type that best describes your data:

[● Delimited] - Characters such as commas or tabs separate each field.

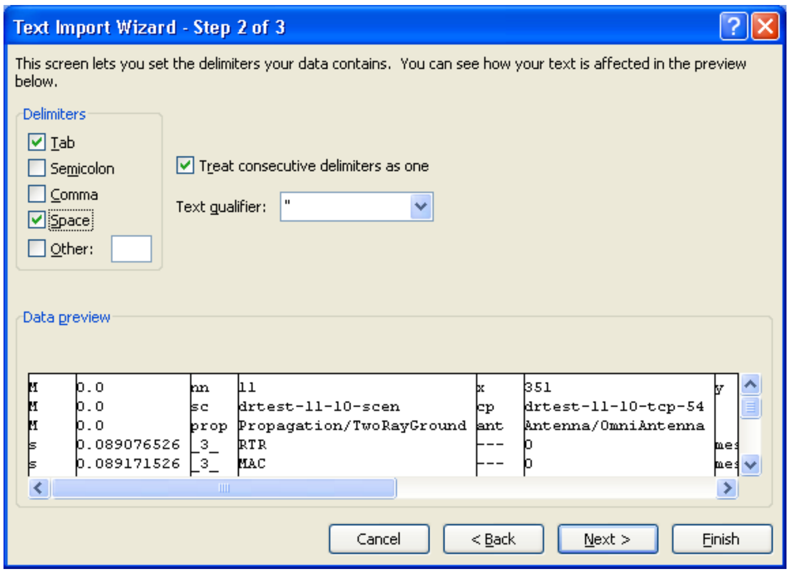
○ Fixed width - Fields are aligned in columns with spaces between each field.

Start import at row: 4 File origin: 437 : OEM United States

Preview of file X:\dave\ns\drtestz-out.tr.

```
1 M 0.0 nn 11 x 351 y 351 rp DSDV
2 M 0.0 sc drtest-11-10-scen cp drtest-11-10-tcp-54
3 M 0.0 prop Propagation/TwoRayGround ant Antenna/OmniAntenna
4 s 0.089076526 _3_ RTR  --- 0 message 32 [0 0 0 0] ------- [3:255 -1:255
5 s 0.089171526 _3_ MAC  --- 0 message 81 [0 ffffffff 3 800] ------- [3:25
```

Figure 6.19: Import text into a spreadsheet



Text Import Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

Delimiters

- [✓] Tab
- [ ] Semicolon
- [ ] Comma
- [✓] Space
- [ ] Other:

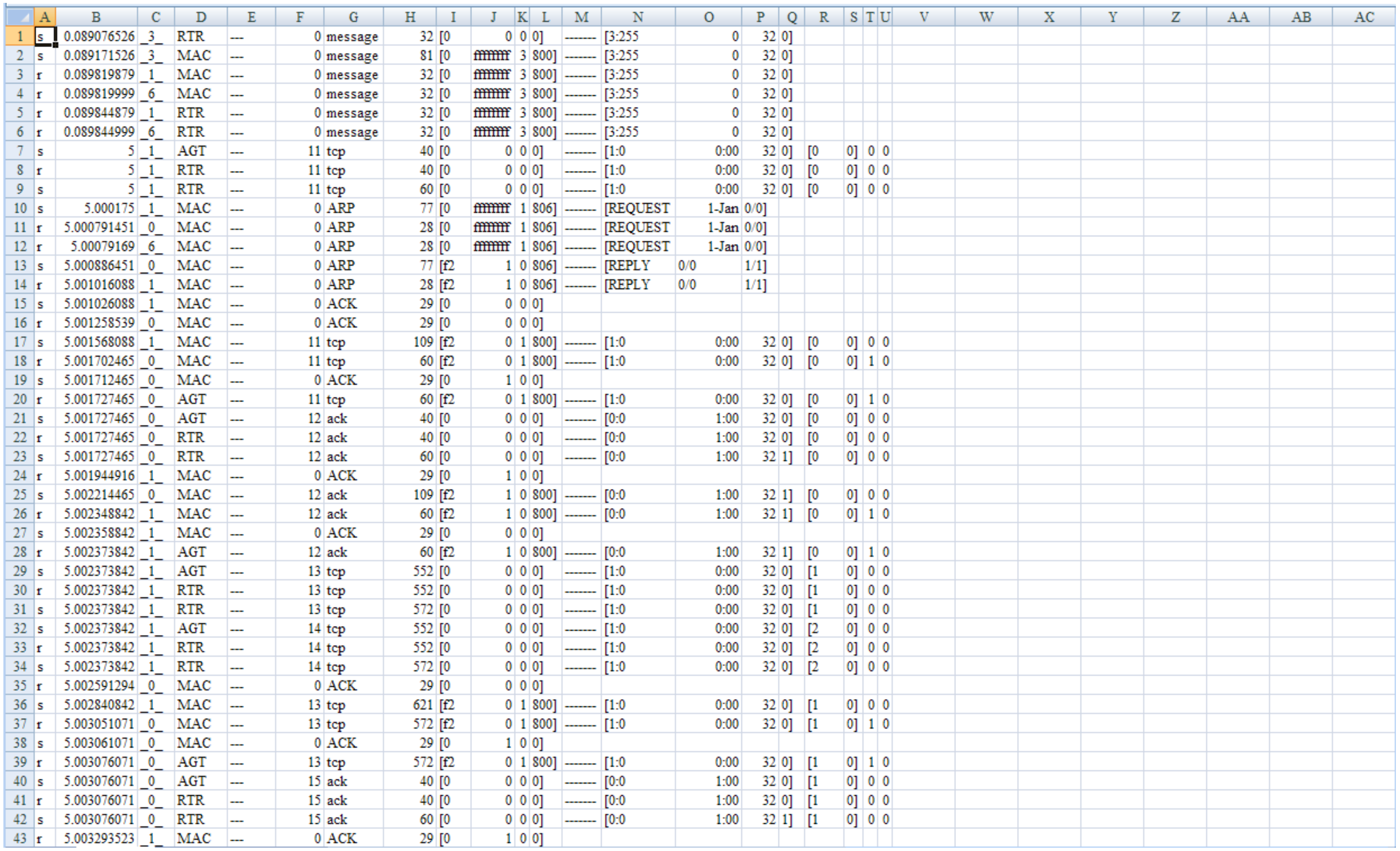
[✓] Treat consecutive delimiters as one

Text qualifier: "

Data preview

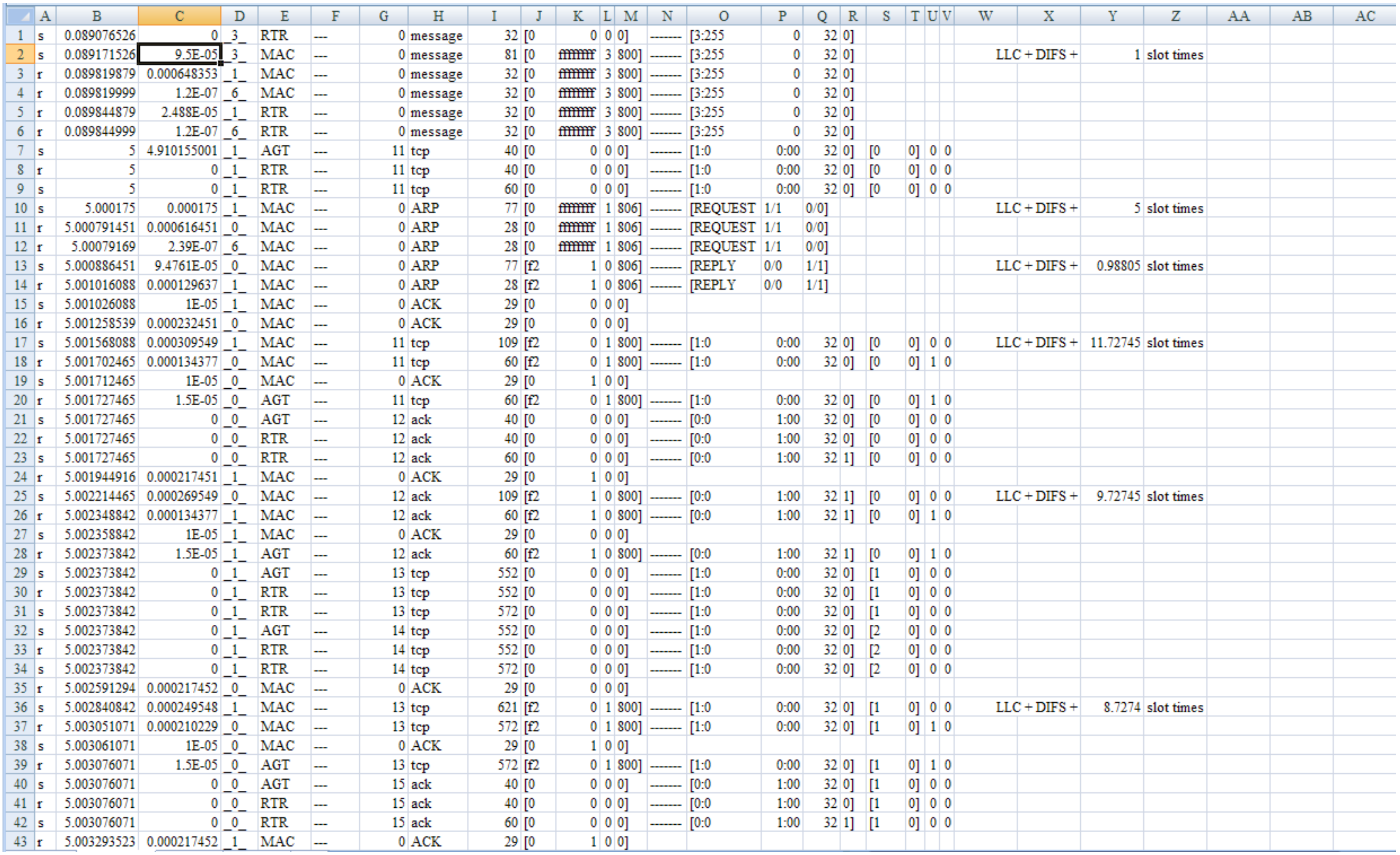
```
M  0.0          nn    11                        x    351                      y
M  0.0          sc    drtest-11-10-scen         cp   drtest-11-10-tcp-54
M  0.0          prop  Propagation/TwoRayGround  ant  Antenna/OmniAntenna
s  0.089076526  _3_   RTR                       ---  0                        me
s  0.089171526  _3_   MAC                       ---  0                        me
```

Figure 6.20: Data delimited by any whitespace



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	s	0.089076526	_3_	RTR	---	0	message	32	[0	0	0	0]	-------	[3:255	0	32	0]												
2	s	0.089171526	_3_	MAC	---	0	message	81	[0	ffffffff	3	800]	-------	[3:255	0	32	0]												
3	r	0.089819879	_1_	MAC	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]												
4	r	0.089819999	_6_	MAC	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]												
5	r	0.089844879	_1_	RTR	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]												
6	r	0.089844999	_6_	RTR	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]												
7	s	5	_1_	AGT	---	11	tcp	40	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0								
8	r	5	_1_	RTR	---	11	tcp	40	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0								
9	s	5	_1_	RTR	---	11	tcp	60	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0								
10	s	5.000175	_1_	MAC	---	0	ARP	77	[0	ffffffff	1	806]	-------	[REQUEST	1-Jan	0/0]													
11	r	5.000791451	_0_	MAC	---	0	ARP	28	[0	ffffffff	1	806]	-------	[REQUEST	1-Jan	0/0]													
12	r	5.00079169	_6_	MAC	---	0	ARP	28	[0	ffffffff	1	806]	-------	[REQUEST	1-Jan	0/0]													
13	s	5.000886451	_0_	MAC	---	0	ARP	77	[f2	1	0	806]	-------	[REPLY	0/0	1/1]													
14	r	5.001016088	_1_	MAC	---	0	ARP	28	[f2	1	0	806]	-------	[REPLY	0/0	1/1]													
15	s	5.001026088	_1_	MAC	---	0	ACK	29	[0	0	0	0]																	
16	r	5.001258539	_0_	MAC	---	0	ACK	29	[0	0	0	0]																	
17	s	5.001568088	_1_	MAC	---	11	tcp	109	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	0	0								
18	r	5.001702465	_0_	MAC	---	11	tcp	60	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	1	0								
19	s	5.001712465	_0_	MAC	---	0	ACK	29	[0	1	0	0]																	
20	r	5.001727465	_0_	AGT	---	11	tcp	60	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	1	0								
21	s	5.001727465	_0_	AGT	---	12	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[0	0]	0	0								
22	r	5.001727465	_0_	RTR	---	12	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[0	0]	0	0								
23	s	5.001727465	_0_	RTR	---	12	ack	60	[0	0	0	0]	-------	[0:0	1:00	32	1]	[0	0]	0	0								
24	r	5.001944916	_1_	MAC	---	0	ACK	29	[0	1	0	0]																	
25	s	5.002214465	_0_	MAC	---	12	ack	109	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	0	0								
26	r	5.002348842	_1_	MAC	---	12	ack	60	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	1	0								
27	s	5.002358842	_1_	MAC	---	0	ACK	29	[0	0	0	0]																	
28	r	5.002373842	_1_	AGT	---	12	ack	60	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	1	0								
29	s	5.002373842	_1_	AGT	---	13	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0								
30	r	5.002373842	_1_	RTR	---	13	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0								
31	s	5.002373842	_1_	RTR	---	13	tcp	572	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0								
32	s	5.002373842	_1_	AGT	---	14	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0								
33	r	5.002373842	_1_	RTR	---	14	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0								
34	s	5.002373842	_1_	RTR	---	14	tcp	572	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0								
35	r	5.002591294	_0_	MAC	---	0	ACK	29	[0	0	0	0]																	
36	s	5.002840842	_1_	MAC	---	13	tcp	621	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	0	0								
37	r	5.003051071	_0_	MAC	---	13	tcp	572	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	1	0								
38	s	5.003061071	_0_	MAC	---	0	ACK	29	[0	1	0	0]																	
39	r	5.003076071	_0_	AGT	---	13	tcp	572	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	1	0								
40	s	5.003076071	_0_	AGT	---	15	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[1	0]	0	0								
41	r	5.003076071	_0_	RTR	---	15	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[1	0]	0	0								
42	s	5.003076071	_0_	RTR	---	15	ack	60	[0	0	0	0]	-------	[0:0	1:00	32	1]	[1	0]	0	0								
43	r	5.003293523	_1_	MAC	---	0	ACK	29	[0	1	0	0]																	

Figure 6.21: Trace output in a spreadsheet



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	s	0.089076526	0	_3_	RTR	---	0	message	32	[0	0	0	0]	-------	[3:255	0	32	0]											
2	s	0.089171526	9.5E-05	_3_	MAC	---	0	message	81	[0	ffffffff	3	800]	-------	[3:255	0	32	0]						LLC + DIFS +	1	slot times			
3	r	0.089819879	0.000648353	_1_	MAC	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]											
4	r	0.089819999	1.2E-07	_6_	MAC	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]											
5	r	0.089844879	2.488E-05	_1_	RTR	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]											
6	r	0.089844999	1.2E-07	_6_	RTR	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]											
7	s	5	4.910155001	_1_	AGT	---	11	tcp	40	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0							
8	r	5	0	_1_	RTR	---	11	tcp	40	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0							
9	s	5	0	_1_	RTR	---	11	tcp	60	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0							
10	s	5.000175	0.000175	_1_	MAC	---	0	ARP	77	[0	ffffffff	1	806]	-------	[REQUEST	1/1	0/0]							LLC + DIFS +	5	slot times			
11	r	5.000791451	0.000616451	_0_	MAC	---	0	ARP	28	[0	ffffffff	1	806]	-------	[REQUEST	1/1	0/0]												
12	r	5.00079169	2.39E-07	_6_	MAC	---	0	ARP	28	[0	ffffffff	1	806]	-------	[REQUEST	1/1	0/0]												
13	s	5.000886451	9.4761E-05	_0_	MAC	---	0	ARP	77	[f2	1	0	806]	-------	[REPLY	0/0	1/1]							LLC + DIFS +	0.98805	slot times			
14	r	5.001016088	0.000129637	_1_	MAC	---	0	ARP	28	[f2	1	0	806]	-------	[REPLY	0/0	1/1]												
15	s	5.001026088	1E-05	_1_	MAC	---	0	ACK	29	[0	0	0	0]																
16	r	5.001258539	0.000232451	_0_	MAC	---	0	ACK	29	[0	0	0	0]																
17	s	5.001568088	0.000309549	_1_	MAC	---	11	tcp	109	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	0	0		LLC + DIFS +	11.72745	slot times			
18	r	5.001702465	0.000134377	_0_	MAC	---	11	tcp	60	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	1	0							
19	s	5.001712465	1E-05	_0_	MAC	---	0	ACK	29	[0	1	0	0]																
20	r	5.001727465	1.5E-05	_0_	AGT	---	11	tcp	60	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	1	0							
21	s	5.001727465	0	_0_	AGT	---	12	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[0	0]	0	0							
22	r	5.001727465	0	_0_	RTR	---	12	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[0	0]	0	0							
23	s	5.001727465	0	_0_	RTR	---	12	ack	60	[0	0	0	0]	-------	[0:0	1:00	32	1]	[0	0]	0	0							
24	r	5.001944916	0.000217451	_1_	MAC	---	0	ACK	29	[0	1	0	0]																
25	s	5.002214465	0.000269549	_0_	MAC	---	12	ack	109	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	0	0		LLC + DIFS +	9.72745	slot times			
26	r	5.002348842	0.000134377	_1_	MAC	---	12	ack	60	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	1	0							
27	s	5.002358842	1E-05	_1_	MAC	---	0	ACK	29	[0	0	0	0]																
28	r	5.002373842	1.5E-05	_1_	AGT	---	12	ack	60	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	1	0							
29	s	5.002373842	0	_1_	AGT	---	13	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0							
30	r	5.002373842	0	_1_	RTR	---	13	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0							
31	s	5.002373842	0	_1_	RTR	---	13	tcp	572	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0							
32	s	5.002373842	0	_1_	AGT	---	14	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0							
33	r	5.002373842	0	_1_	RTR	---	14	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0							
34	s	5.002373842	0	_1_	RTR	---	14	tcp	572	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0							
35	r	5.002591294	0.000217452	_0_	MAC	---	0	ACK	29	[0	0	0	0]																
36	s	5.002840842	0.000249548	_1_	MAC	---	13	tcp	621	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	0	0		LLC + DIFS +	8.7274	slot times			
37	r	5.003051071	0.000210229	_0_	MAC	---	13	tcp	572	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	1	0							
38	s	5.003061071	1E-05	_0_	MAC	---	0	ACK	29	[0	1	0	0]																
39	r	5.003076071	1.5E-05	_0_	AGT	---	13	tcp	572	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	1	0							
40	s	5.003076071	0	_0_	AGT	---	15	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[1	0]	0	0							
41	r	5.003076071	0	_0_	RTR	---	15	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[1	0]	0	0							
42	s	5.003076071	0	_0_	RTR	---	15	ack	60	[0	0	0	0]	-------	[0:0	1:00	32	1]	[1	0]	0	0							
43	r	5.003293523	0.000217452	_1_	MAC	---	0	ACK	29	[0	1	0	0]																

Figure 6.22: Inclusion of the initial formulae in the trace output

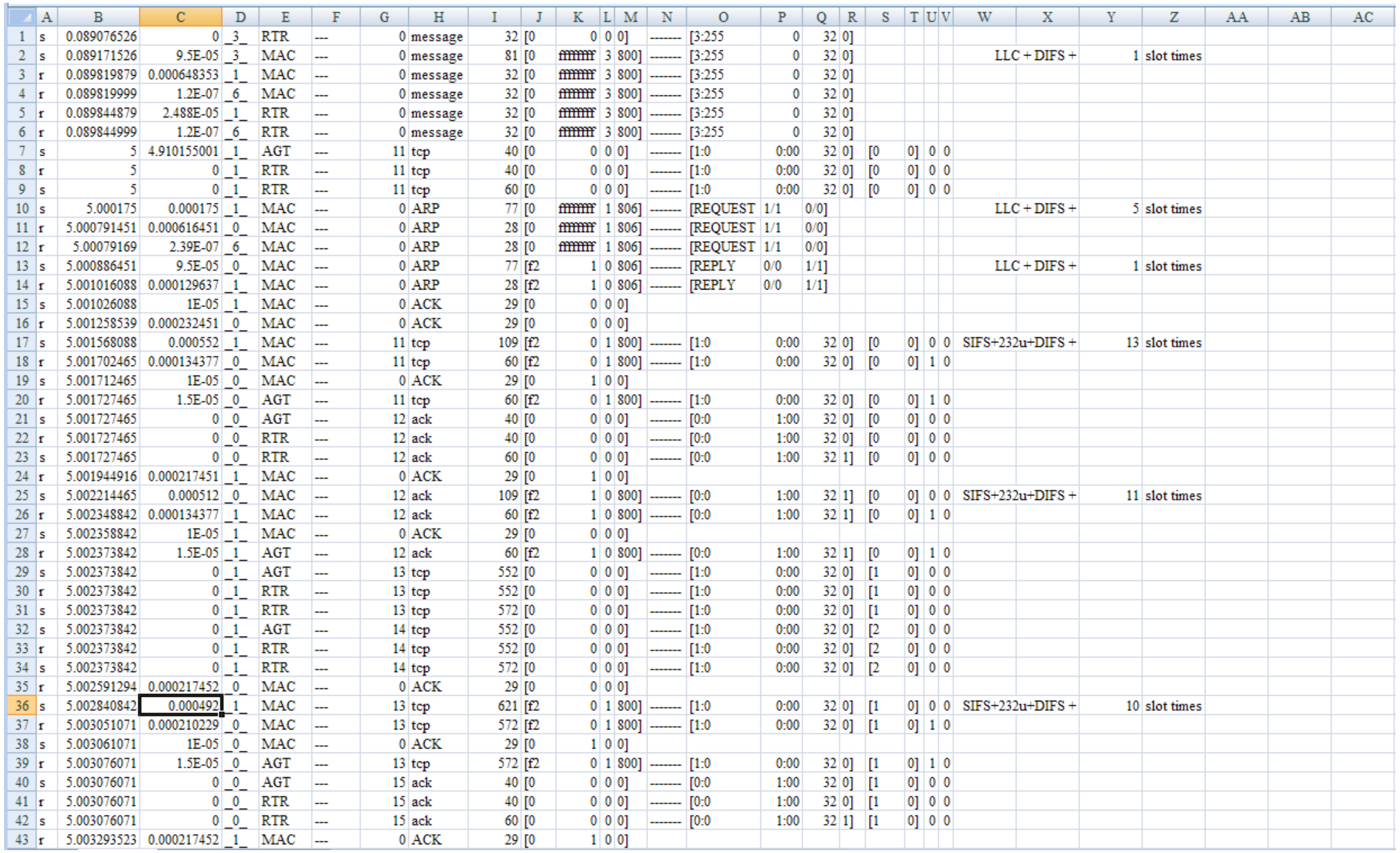
Throughout the first five seconds of the simulation, for over 200 trace entries (only a few are shown in the graphic of Figure 6.22), where only single STAs were transmitting broadcast packets (numerous route messages and an ARP request), these formulae worked well. However, as can be seen starting at row 13 in the example of Figure 6.22, as soon as unicast transmissions started, these formulae were insufficient for a number of reasons.

The first issue was because the timers were no longer related to the previous event in the trace file. Up until this point, the channel was quiescent and the previous event was the routing subsystem passing the frame to the MAC subsystem on the same node. Once unicast transmissions started, the previous event in the trace file was typically not relevant to the timing of the pending transmission. This was rectified by amending the errant formulae from $C_x = B_x - B_{x-1}$ to $C_{send} = B_{send} - B_{lastframereceivedbythisnode}$. In the majority of cases, two distinct patterns were evident in the trace:

- A MAC subsystem would receive a frame, send a MAC ACK, pass the frame to the TCP agent, the agent creates the next frame (a TCP ACK for a TCP data frame or a TCP data frame after a TCP ACK) and sends it to the routing subsystem, the routing subsystem receives the next frame, the routing subsystem addresses the next frame and sends it to the MAC subsystem, and the MAC subsystem sends the next queued frame. In this case, the formula for the third column was changed from $C_x = B_x - B_{x-1}$ to $C_{send} = B_{send} - B_{send-7}$.

- A MAC subsystem would receive a frame (TCP ACK), send a MAC ACK, pass the frame to the TCP agent, the agent creates the next TCP data frame and sends it to the routing subsystem, the routing subsystem receives the next frame, the routing subsystem addresses the next frame and sends it to the MAC subsystem, the TCP agent expands the window and creates a second TCP data frame and sends it to the routing subsystem, the routing subsystem receives the second frame, the routing subsystem addresses the second frame and sends it to the MAC subsystem, and the MAC subsystem sends the next queued frame. In this case, the formula for the third column was changed from $C_x = B_x - B_{x-1}$ to $C_{send} = B_{send} - B_{send-10}$.

The second issue was the calculation of slot times was now incorrect, even with the corrected elapsed time values, as there was no longer only broadcast



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	s	0.089076526	0	_3_	RTR	---	0	message	32	[0	0	0	0]	-------	[3:255	0	32	0]											
2	s	0.089171526	9.5E-05	_3_	MAC	---	0	message	81	[0	ffffffff	3	800]	-------	[3:255	0	32	0]						LLC + DIFS +	1	slot times			
3	r	0.089819879	0.000648353	_1_	MAC	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]											
4	r	0.089819999	1.2E-07	_6_	MAC	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]											
5	r	0.089844879	2.488E-05	_1_	RTR	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]											
6	r	0.089844999	1.2E-07	_6_	RTR	---	0	message	32	[0	ffffffff	3	800]	-------	[3:255	0	32	0]											
7	s	5	4.910155001	_1_	AGT	---	11	tcp	40	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0							
8	r	5	0	_1_	RTR	---	11	tcp	40	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0							
9	s	5	0	_1_	RTR	---	11	tcp	60	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0							
10	s	5.000175	0.000175	_1_	MAC	---	0	ARP	77	[0	ffffffff	1	806]	-------	[REQUEST	1/1	0/0]							LLC + DIFS +	5	slot times			
11	r	5.000791451	0.000616451	_0_	MAC	---	0	ARP	28	[0	ffffffff	1	806]	-------	[REQUEST	1/1	0/0]												
12	r	5.00079169	2.39E-07	_6_	MAC	---	0	ARP	28	[0	ffffffff	1	806]	-------	[REQUEST	1/1	0/0]												
13	s	5.000886451	9.5E-05	_0_	MAC	---	0	ARP	77	[f2	1	0	806]	-------	[REPLY	0/0	1/1]							LLC + DIFS +	1	slot times			
14	r	5.001016088	0.000129637	_1_	MAC	---	0	ARP	28	[f2	1	0	806]	-------	[REPLY	0/0	1/1]												
15	s	5.001026088	1E-05	_1_	MAC	---	0	ACK	29	[0	0	0	0]																
16	r	5.001258539	0.000232451	_0_	MAC	---	0	ACK	29	[0	0	0	0]																
17	s	5.001568088	0.000552	_1_	MAC	---	11	tcp	109	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	0	0	SIFS+232u+DIFS +		13	slot times			
18	r	5.001702465	0.000134377	_0_	MAC	---	11	tcp	60	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	1	0							
19	s	5.001712465	1E-05	_0_	MAC	---	0	ACK	29	[0	1	0	0]																
20	r	5.001727465	1.5E-05	_0_	AGT	---	11	tcp	60	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	1	0							
21	s	5.001727465	0	_0_	AGT	---	12	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[0	0]	0	0							
22	r	5.001727465	0	_0_	RTR	---	12	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[0	0]	0	0							
23	s	5.001727465	0	_0_	RTR	---	12	ack	60	[0	0	0	0]	-------	[0:0	1:00	32	1]	[0	0]	0	0							
24	r	5.001944916	0.000217451	_1_	MAC	---	0	ACK	29	[0	1	0	0]																
25	s	5.002214465	0.000512	_0_	MAC	---	12	ack	109	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	0	0	SIFS+232u+DIFS +		11	slot times			
26	r	5.002348842	0.000134377	_1_	MAC	---	12	ack	60	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	1	0							
27	s	5.002358842	1E-05	_1_	MAC	---	0	ACK	29	[0	0	0	0]																
28	r	5.002373842	1.5E-05	_1_	AGT	---	12	ack	60	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	1	0							
29	s	5.002373842	0	_1_	AGT	---	13	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0							
30	r	5.002373842	0	_1_	RTR	---	13	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0							
31	s	5.002373842	0	_1_	RTR	---	13	tcp	572	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0							
32	s	5.002373842	0	_1_	AGT	---	14	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0							
33	r	5.002373842	0	_1_	RTR	---	14	tcp	552	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0							
34	s	5.002373842	0	_1_	RTR	---	14	tcp	572	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0							
35	r	5.002591294	0.000217452	_0_	MAC	---	0	ACK	29	[0	0	0	0]																
36	s	5.002840842	0.000492	_1_	MAC	---	13	tcp	621	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	0	0	SIFS+232u+DIFS +		10	slot times			
37	r	5.003051071	0.000210229	_0_	MAC	---	13	tcp	572	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	1	0							
38	s	5.003061071	1E-05	_0_	MAC	---	0	ACK	29	[0	1	0	0]																
39	r	5.003076071	1.5E-05	_0_	AGT	---	13	tcp	572	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	1	0							
40	s	5.003076071	0	_0_	AGT	---	15	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[1	0]	0	0							
41	r	5.003076071	0	_0_	RTR	---	15	ack	40	[0	0	0	0]	-------	[0:0	1:00	32	0]	[1	0]	0	0							
42	s	5.003076071	0	_0_	RTR	---	15	ack	60	[0	0	0	0]	-------	[0:0	1:00	32	1]	[1	0]	0	0							
43	r	5.003293523	0.000217452	_1_	MAC	---	0	ACK	29	[0	1	0	0]																

Figure 6.23: Formulae corrections for directed traffic

traffic on a quiescent channel and so *the LLC processing delay was no longer relevant to the elapsed time before transmission begins.* This single observation is later pivotal in explaining the effect of encryption overhead on throughput.

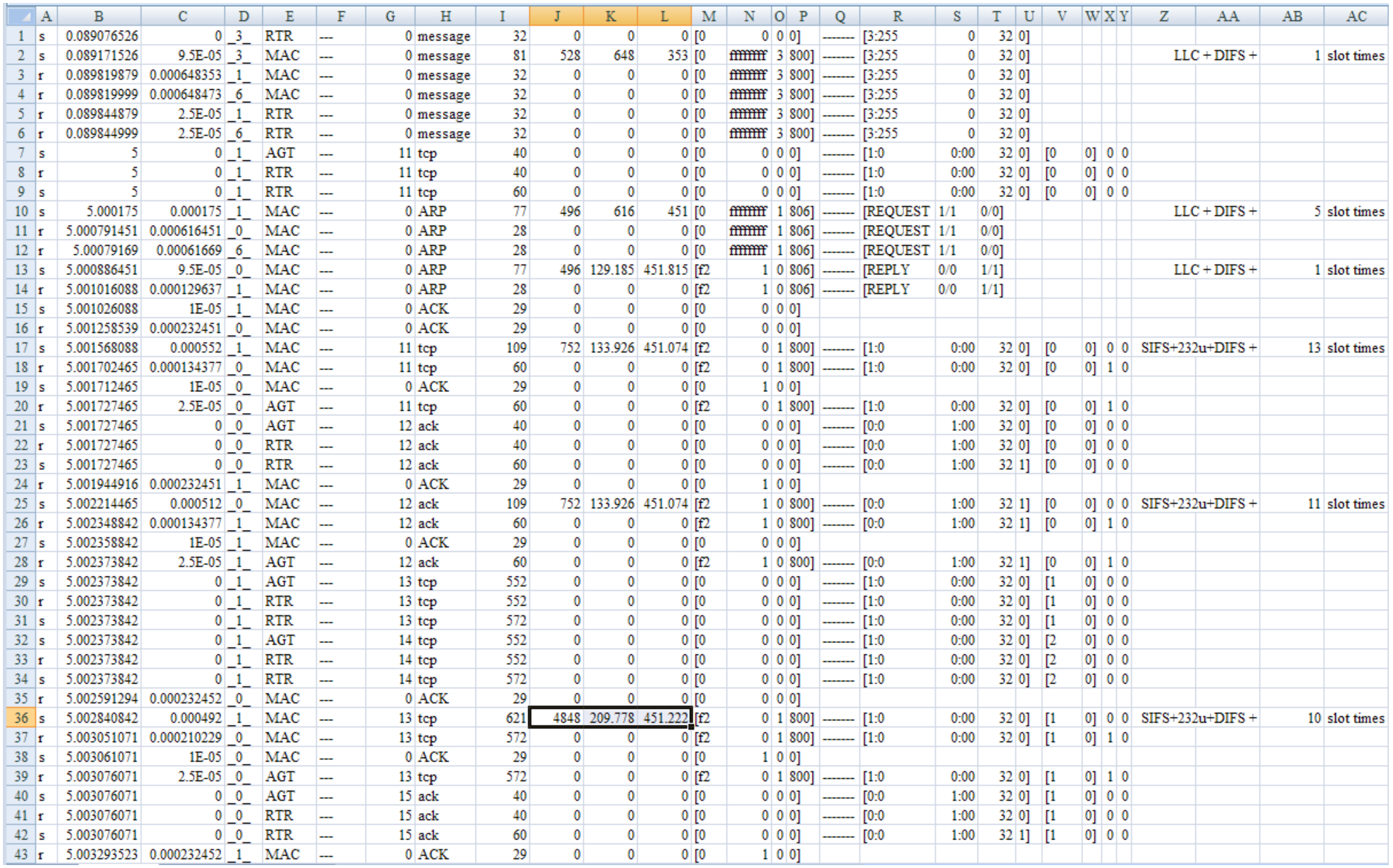
In this case, the LLC processing delay was always 25 µs, however on receiving directed traffic (a unicast frame), a STA must wait a SIFS and send a MAC-layer ACK back to the sender. This takes 10 µs and 232 µs, a total of 242 µs, before the STA can begin its DIFS and backoff, making the 25 µs LLC processing delay irrelevant here.

For directed traffic, the number of slots in the backoff was now determined from the elapsed time by subtracting a SIFS and the time to transmit a MAC-layer ACK and a DIFS, being $10 + 232 + 50 = 292$ µs, and dividing the result by the slot time, changing the formula from $Y_x = (C_x - 0.000075)/0.00002$ to $Y_x = (C_x - 0.000292)/0.00002$, as shown in Figure 6.23.

Further enhancements were to insert three more columns after the packet size in bytes, containing, for each transmitted MAC data frame, the MPDU size in bits, the time to transmit the frame and the time to propagate the signal to the receiver, as shown in Figure 6.24. The frames of interest were those with 'on-the-wire' sizes (for *ns-2* with DSDV ad-hoc routing) of 81 bytes (routing broadcast advertisements), 77 bytes (ARP requests and replies), 109 bytes (empty TCP SYN or ACK), or 621 bytes (TCP data packets).

Where the event packet size was one of 81, 77, 109 or 621, the MPDU size was calculated by multiplying the packet size in bytes by 8 bits and subtracting the 120 bits of PLCP short preamble and PLCP header. Otherwise this field was set to zero, as it was not an outgoing packet from the MAC layer, other than a MAC ACK, which is 29 bytes either sending or receiving (the headers are not 'stripped' on receipt as the MAC sub-layer does not pass it to a higher layer). This was achieved using the formula IF (bytes=81 OR bytes=77 OR bytes=109 OR bytes=621) THEN value=bytes*8-120 ELSE value=0. It should be noted that the value is the size of the MPDU for those frames of interest, but a value of zero does not necessarily mean an empty frame — just not a frame of interest here. The formula used was:
$J_x = IF(OR(I_x = 81, I_x = 77, I_x = 109, I_x = 621), I_x * 8 - 120, 0)$

The time to transmit the frame, for those frames for which the MPDU size has been calculated, depended on whether or not the frame was broadcast/multicast or unicast. The preamble of all frames is always sent at 1 Mbps. The entire



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	s	0.089076526	0	_3_	RTR	---	0	message	32	0	0	0	[0	0	0	0]	-------	[3:255	0	32	0]								
2	s	0.089171526	9.5E-05	_3_	MAC	---	0	message	81	528	648	353	[0	fffffff	3	800]	-------	[3:255	0	32	0]						LLC + DIFS +	1	slot times
3	r	0.089819879	0.000648353	_1_	MAC	---	0	message	32	0	0	0	[0	fffffff	3	800]	-------	[3:255	0	32	0]								
4	r	0.089819999	0.000648473	_6_	MAC	---	0	message	32	0	0	0	[0	fffffff	3	800]	-------	[3:255	0	32	0]								
5	r	0.089844879	2.5E-05	_1_	RTR	---	0	message	32	0	0	0	[0	fffffff	3	800]	-------	[3:255	0	32	0]								
6	r	0.089844999	2.5E-05	_6_	RTR	---	0	message	32	0	0	0	[0	fffffff	3	800]	-------	[3:255	0	32	0]								
7	s	5	0	_1_	AGT	---	11	tcp	40	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0				
8	r	5	0	_1_	RTR	---	11	tcp	40	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0				
9	s	5	0	_1_	RTR	---	11	tcp	60	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[0	0]	0	0				
10	s	5.000175	0.000175	_1_	MAC	---	0	ARP	77	496	616	451	[0	fffffff	1	806]	-------	[REQUEST	1/1	0/0]							LLC + DIFS +	5	slot times
11	r	5.000791451	0.000616451	_0_	MAC	---	0	ARP	28	0	0	0	[0	fffffff	1	806]	-------	[REQUEST	1/1	0/0]									
12	r	5.00079169	0.00061669	_6_	MAC	---	0	ARP	28	0	0	0	[0	fffffff	1	806]	-------	[REQUEST	1/1	0/0]									
13	s	5.000886451	9.5E-05	_0_	MAC	---	0	ARP	77	496	129.185	451.815	[f2	1	0	806]	-------	[REPLY	0/0	1/1]							LLC + DIFS +	1	slot times
14	r	5.001016088	0.000129637	_1_	MAC	---	0	ARP	28	0	0	0	[f2	1	0	806]	-------	[REPLY	0/0	1/1]									
15	s	5.001026088	1E-05	_1_	MAC	---	0	ACK	29	0	0	0	[0	0	0	0]													
16	r	5.001258539	0.000232451	_0_	MAC	---	0	ACK	29	0	0	0	[0	0	0	0]													
17	s	5.001568088	0.000552	_1_	MAC	---	11	tcp	109	752	133.926	451.074	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	0	0	SIFS+232u+DIFS +		13	slot times
18	r	5.001702465	0.000134377	_0_	MAC	---	11	tcp	60	0	0	0	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	1	0				
19	s	5.001712465	1E-05	_0_	MAC	---	0	ACK	29	0	0	0	[0	1	0	0]													
20	r	5.001727465	2.5E-05	_0_	AGT	---	11	tcp	60	0	0	0	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[0	0]	1	0				
21	s	5.001727465	0	_0_	AGT	---	12	ack	40	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	0]	[0	0]	0	0				
22	r	5.001727465	0	_0_	RTR	---	12	ack	40	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	0]	[0	0]	0	0				
23	s	5.001727465	0	_0_	RTR	---	12	ack	60	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	1]	[0	0]	0	0				
24	r	5.001944916	0.000232451	_1_	MAC	---	0	ACK	29	0	0	0	[0	1	0	0]													
25	s	5.002214465	0.000512	_0_	MAC	---	12	ack	109	752	133.926	451.074	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	0	0	SIFS+232u+DIFS +		11	slot times
26	r	5.002348842	0.000134377	_1_	MAC	---	12	ack	60	0	0	0	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	1	0				
27	s	5.002358842	1E-05	_1_	MAC	---	0	ACK	29	0	0	0	[0	0	0	0]													
28	r	5.002373842	2.5E-05	_1_	AGT	---	12	ack	60	0	0	0	[f2	1	0	800]	-------	[0:0	1:00	32	1]	[0	0]	1	0				
29	s	5.002373842	0	_1_	AGT	---	13	tcp	552	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0				
30	r	5.002373842	0	_1_	RTR	---	13	tcp	552	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0				
31	s	5.002373842	0	_1_	RTR	---	13	tcp	572	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[1	0]	0	0				
32	s	5.002373842	0	_1_	AGT	---	14	tcp	552	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0				
33	r	5.002373842	0	_1_	RTR	---	14	tcp	552	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0				
34	s	5.002373842	0	_1_	RTR	---	14	tcp	572	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[2	0]	0	0				
35	r	5.002591294	0.000232452	_0_	MAC	---	0	ACK	29	0	0	0	[0	0	0	0]													
36	s	5.002840842	0.000492	_1_	MAC	---	13	tcp	621	4848	209.778	451.222	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	0	0	SIFS+232u+DIFS +		10	slot times
37	r	5.003051071	0.000210229	_0_	MAC	---	13	tcp	572	0	0	0	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	1	0				
38	s	5.003061071	1E-05	_0_	MAC	---	0	ACK	29	0	0	0	[0	1	0	0]													
39	r	5.003076071	2.5E-05	_0_	AGT	---	13	tcp	572	0	0	0	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[1	0]	1	0				
40	s	5.003076071	0	_0_	AGT	---	15	ack	40	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	0]	[1	0]	0	0				
41	r	5.003076071	0	_0_	RTR	---	15	ack	40	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	0]	[1	0]	0	0				
42	s	5.003076071	0	_0_	RTR	---	15	ack	60	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	1]	[1	0]	0	0				
43	r	5.003293523	0.000232452	_1_	MAC	---	0	ACK	29	0	0	0	[0	1	0	0]													

Figure 6.24: Addition of MSDU size, transmit and propagation times

of broadcast/multicast frames are always sent at a Basic Rate, also 1 Mbps in these simulations. The PLCP header of directed frames should be sent at 1 Mbps with the standard (long) preamble or 2 Mbps with the short preamble used here. However, ns-2 always sends the PLCP preamble and PLCP header together at 1 Mbps, taking 120 µs. The remainder of directed frames is sent at the Data Rate of 54 Mbps.

Thus, the formula to determine the time to transmit the frame, in microseconds, was IF msdubits=0 THEN value=0 ELSE IF destination=ffffffff THEN value=bytes*8 ELSE value=120+msdubits/54. The formula used was:
$K_x = IF(J_x = 0, 0, IF(N_x = \text{“}ffffffff\text{”}, I_x * 8, 120 + J_x/54))$

The time to propagate the signal from source to destination, for those frames for which the MPDU size has been calculated, in nanoseconds, was determined from the elapsed time, in seconds, between the send event and the next corresponding receive event. Note that *ns-2* traces record the *start* of a send event, the time of the transmission of the leading edge of the start of the signal, and the *completion* of a receive event, when the last component of the signal has been successfully received. as such, at the MAC sub-layer, the difference between these two times is the time to transmit the frame plus the time for the signal to propagate from source to destination, in seconds.

The formula to determine the time to propagate the signal in nanoseconds, from the elapsed time in seconds and the transmission time in microseconds, was IF txtime=0 THEN value=0 ELSE value=elapsedtime*1000000000-txtime*1000. The formula used was: $L_x = IF(K_x = 0, 0, C_{x+1} * 1000000000 - K_x * 1000)$

In addition, the elapsed times for receiving a MAC ACK were changed from $C_x = B_x - B_{x-1}$ to $C_{recvACK} = B_{recvACK} - B_{sendACK}$, to display an elapsed time of 0.000232nnn, where the 232 microseconds is the transmission time for a MAC ACK and the ‘nnn’ nanoseconds is the propagation time for the signal to travel from the transmitting node to the receiving node.

6.7.1 Continuing Development

As the analysis continued further into the simulations, these formulae were enhanced to account for more and more different combinations of events. Later enhancements accounted for interactions between multiple STA, with elucidation of inter-STA propagation delays and their effects on transmission contention delays.

The formulae for the third column, for each MAC sub-layer transmit event at time x (after a MAC ACK receive at time $x-1$) became:

```
IF mac_ack_recv_from_same_node THEN elapsedtime=current-mac_ack_recv
ELSE IF last_data_recv_by_same_node THEN elapsedtime=current-last_data_recv
        ELSE elapsedtime=current-mac_ack_send
```

Encoded as shown in Figure 6.25:

...

$C_n = B_n - B_{n-1}$

...

$C_{x-1} = B_{x-1} - B_{x-6}$

$C_x = IF(D_{x-1} = D_x, B_x - B_{x-1}, IF(D_{x-7} = D_x, B_x - B_{x-7}, B_x - B_{x-6}))$

$C_{x+1} = B_{x+1} - B_x$

$C_{x+2} = B_{x+2} - B_{x+1}$

$C_{x+3} = B_{x+3} - B_{x+1}$

$C_n = B_n - B_{n-1}$

...

The description for the third-last column became:

```
IF mac_ack_recv_from_same_node THEN description="RxACK + DIFS +" (slots)
ELSE IF last_data_recv_by_same_node THEN description="Rx+SIFS+ACK+DIFS+"
        ELSE description="RxACK + DIFS +"
```

and the formula to calculate the slot times became:

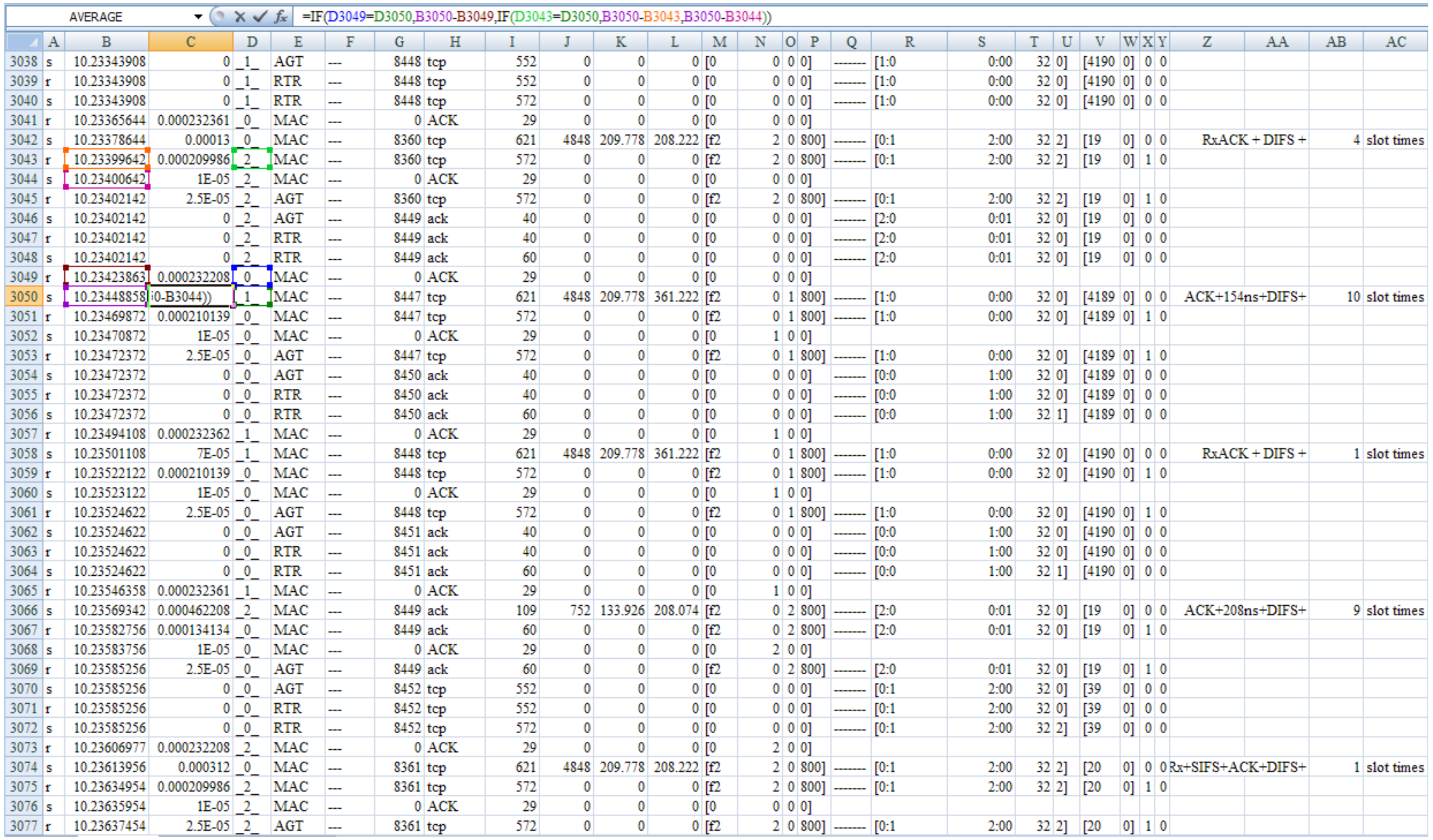
```
IF mac_ack_recv_from_same_node THEN slots=(elapsed-difs)/slottime
ELSE IF last_data_recv_by_same_node THEN slots=(elapsed-(sifs+ack+difs))/slottime
        ELSE slots=(elapsed-(ack+difs)-propagation)/slottime
```

Encoded as:

$AA_x = IF(D_{x-1} = D_x, \text{“}RxACK + DIFS+\text{”},$
$\quad IF(D_{x-7} = D_x, \text{“}Rx + SIFS + ACK + DIFS+\text{”}, \text{“}ACK+\text{”}\&$
$\quad TEXT((C_x - INT(C_x * 1000000)/1000000) * 1000000000, 0)\&\text{“}ns + DIFS+\text{”}))$

and (in each case set to zero if very small):

$AB_x = IF(D_{x-1} = D_x, IF((C_x - 0.00005)/0.00002 < 0.1, 0, (C_x - 0.00005)/0.00002),$
$\quad IF(D_{x-7} = D_x, IF((C_x - 0.000292)/0.00002 < 0.1, 0, (C_x - 0.000292)/0.00002),$
$\quad (C_x - 0.000282 - (C_x - INT(C_x * 1000000)/1000000))/0.00002))$



AVERAGE =IF(D3049=D3050,B3050-B3049,IF(D3043=D3050,B3050-B3043,B3050-B3044))

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
3038	s	10.23343908	0	_1_	AGT	---	8448	tcp	552	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[4190	0]	0	0				
3039	r	10.23343908	0	_1_	RTR	---	8448	tcp	552	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[4190	0]	0	0				
3040	s	10.23343908	0	_1_	RTR	---	8448	tcp	572	0	0	0	[0	0	0	0]	-------	[1:0	0:00	32	0]	[4190	0]	0	0				
3041	r	10.23365644	0.000232361	_0_	MAC	---	0	ACK	29	0	0	0	[0	0	0	0]													
3042	s	10.23378644	0.00013	_0_	MAC	---	8360	tcp	621	4848	209.778	208.222	[f2	2	0	800]	-------	[0:1	2:00	32	2]	[19	0]	0	0	RxACK + DIFS +		4	slot times
3043	r	10.23399642	0.000209986	_2_	MAC	---	8360	tcp	572	0	0	0	[f2	2	0	800]	-------	[0:1	2:00	32	2]	[19	0]	1	0				
3044	s	10.23400642	1E-05	_2_	MAC	---	0	ACK	29	0	0	0	[0	0	0	0]													
3045	r	10.23402142	2.5E-05	_2_	AGT	---	8360	tcp	572	0	0	0	[f2	2	0	800]	-------	[0:1	2:00	32	2]	[19	0]	1	0				
3046	s	10.23402142	0	_2_	AGT	---	8449	ack	40	0	0	0	[0	0	0	0]	-------	[2:0	0:01	32	0]	[19	0]	0	0				
3047	r	10.23402142	0	_2_	RTR	---	8449	ack	40	0	0	0	[0	0	0	0]	-------	[2:0	0:01	32	0]	[19	0]	0	0				
3048	s	10.23402142	0	_2_	RTR	---	8449	ack	60	0	0	0	[0	0	0	0]	-------	[2:0	0:01	32	0]	[19	0]	0	0				
3049	r	10.23423863	0.000232208	_0_	MAC	---	0	ACK	29	0	0	0	[0	0	0	0]													
3050	s	10.23448858	0-B3044))	_1_	MAC	---	8447	tcp	621	4848	209.778	361.222	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[4189	0]	0	0	ACK+154ns+DIFS+		10	slot times
3051	r	10.23469872	0.000210139	_0_	MAC	---	8447	tcp	572	0	0	0	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[4189	0]	1	0				
3052	s	10.23470872	1E-05	_0_	MAC	---	0	ACK	29	0	0	0	[0	1	0	0]													
3053	r	10.23472372	2.5E-05	_0_	AGT	---	8447	tcp	572	0	0	0	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[4189	0]	1	0				
3054	s	10.23472372	0	_0_	AGT	---	8450	ack	40	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	0]	[4189	0]	0	0				
3055	r	10.23472372	0	_0_	RTR	---	8450	ack	40	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	0]	[4189	0]	0	0				
3056	s	10.23472372	0	_0_	RTR	---	8450	ack	60	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	1]	[4189	0]	0	0				
3057	r	10.23494108	0.000232362	_1_	MAC	---	0	ACK	29	0	0	0	[0	1	0	0]													
3058	s	10.23501108	7E-05	_1_	MAC	---	8448	tcp	621	4848	209.778	361.222	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[4190	0]	0	0	RxACK + DIFS +		1	slot times
3059	r	10.23522122	0.000210139	_0_	MAC	---	8448	tcp	572	0	0	0	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[4190	0]	1	0				
3060	s	10.23523122	1E-05	_0_	MAC	---	0	ACK	29	0	0	0	[0	1	0	0]													
3061	r	10.23524622	2.5E-05	_0_	AGT	---	8448	tcp	572	0	0	0	[f2	0	1	800]	-------	[1:0	0:00	32	0]	[4190	0]	1	0				
3062	s	10.23524622	0	_0_	AGT	---	8451	ack	40	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	0]	[4190	0]	0	0				
3063	r	10.23524622	0	_0_	RTR	---	8451	ack	40	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	0]	[4190	0]	0	0				
3064	s	10.23524622	0	_0_	RTR	---	8451	ack	60	0	0	0	[0	0	0	0]	-------	[0:0	1:00	32	1]	[4190	0]	0	0				
3065	r	10.23546358	0.000232361	_1_	MAC	---	0	ACK	29	0	0	0	[0	1	0	0]													
3066	s	10.23569342	0.000462208	_2_	MAC	---	8449	ack	109	752	133.926	208.074	[f2	0	2	800]	-------	[2:0	0:01	32	0]	[19	0]	0	0	ACK+208ns+DIFS+		9	slot times
3067	r	10.23582756	0.000134134	_0_	MAC	---	8449	ack	60	0	0	0	[f2	0	2	800]	-------	[2:0	0:01	32	0]	[19	0]	1	0				
3068	s	10.23583756	1E-05	_0_	MAC	---	0	ACK	29	0	0	0	[0	2	0	0]													
3069	r	10.23585256	2.5E-05	_0_	AGT	---	8449	ack	60	0	0	0	[f2	0	2	800]	-------	[2:0	0:01	32	0]	[19	0]	1	0				
3070	s	10.23585256	0	_0_	AGT	---	8452	tcp	552	0	0	0	[0	0	0	0]	-------	[0:1	2:00	32	0]	[39	0]	0	0				
3071	r	10.23585256	0	_0_	RTR	---	8452	tcp	552	0	0	0	[0	0	0	0]	-------	[0:1	2:00	32	0]	[39	0]	0	0				
3072	s	10.23585256	0	_0_	RTR	---	8452	tcp	572	0	0	0	[0	0	0	0]	-------	[0:1	2:00	32	2]	[39	0]	0	0				
3073	r	10.23606977	0.000232208	_2_	MAC	---	0	ACK	29	0	0	0	[0	2	0	0]													
3074	s	10.23613956	0.000312	_0_	MAC	---	8361	tcp	621	4848	209.778	208.222	[f2	2	0	800]	-------	[0:1	2:00	32	2]	[20	0]	0	0	Rx+SIFS+ACK+DIFS+		1	slot times
3075	r	10.23634954	0.000209986	_2_	MAC	---	8361	tcp	572	0	0	0	[f2	2	0	800]	-------	[0:1	2:00	32	2]	[20	0]	1	0				
3076	s	10.23635954	1E-05	_2_	MAC	---	0	ACK	29	0	0	0	[0	0	0	0]													
3077	r	10.23637454	2.5E-05	_2_	AGT	---	8361	tcp	572	0	0	0	[f2	2	0	800]	-------	[0:1	2:00	32	2]	[20	0]	1	0				

Figure 6.25: Enhanced formulae and deterministic text cues

Chapter 7

Testing the Thesis

This chapter details the actual tests undertaken, establishing the control data for UDP and TCP traffic, determining the average throughput and the effects of multi-STA contention. The delays due to normal operation of IEEE 802.11 networks requiring acknowledged directed traffic and the default Distributed Coordination Function (DCF) backoff and their effects on network performance are determined, in particular, the effects of contention with two STAs transmitting simultaneously and the race condition that ensues and the deterministic resolution from the tools being used are examined. This analysis includes the effects where one STA is acting as an Access Point (AP) and thereby dealing with all of the other traffic, either transmitting or receiving data, or acknowledgements of transmitted or received data, as appropriate. Then the proposals are tested, first detailing how the tools were modified to implement the modified protocols.

Initially, dynamically loadable libraries were considered as a way of providing the alternate protocols within the simulator. However, the impracticalities of this are described and this approach is abandoned in favour of simply modifying the simulator source directly.

The initial results were not as expected and the tools were then modified to ensure that this was not due to inaccuracy within the simulation itself. However, results after the additional modifications merely confirmed the initial results and the tests were continued to forge a comprehensive range of excessive traffic situations under both WLS and PWLS. These results are then analysed and discussed in detail.

7.1 Control Data

As a result of the initial investigations in Chapter 6 [section 6.4], the baseline was chosen with 2048-byte (at the source) packets, being produced at 16 Mbps (effectively 2068-byte packets at 16.16 Mbps due to *ns-2* DSDV routing nuances), passed to the MAC layer with a maximum 2304-byte MSDU, via the LLC with a 25 µs overhead, being framed with a short 72-bit preamble and transmitted with a data rate of 54 Mbps and no RTS/CTS exchanges.

Figure 6.18 provides stationary reference data for the chosen baseline control under these conditions, with stationary STA at 1 m range. To provide baseline data for STA involving movement throughout the simulation radio range, *drtestg.tcl* was configured as follows:

```
set val(x)        351                          ;# X dimension of the topo
set val(y)        351                          ;# Y dimension of the topo
set val(sc)       "drtest-2-1-scen"            ;# scene movement file
set val(bps)      16.0                         ;# cbr Mbit rate
set val(stop)     60.0                         ;# simulation time

Agent/UDP set packetSize_ 2304
LL set delay_ 25us
Mac/802_11 set dataRate_ 54Mb
Mac/802_11 set RTSThreshold_ 3000
Mac/802_11 set PreambleLength_ 72

for {set i 1} {$i < $val(nn) } {incr i} {
    set udp_($i) [new Agent/UDP]
    $ns_ attach-agent $node_($i) $udp_($i)
    set sink_($i) [new Agent/LossMonitor]
    $ns_ attach-agent $node_(0) $sink_($i)
    set cbr_($i) [new Application/Traffic/CBR]
    $cbr_($i) set packetSize_ 2048
    $cbr_($i) set interval_ [expr 0.016384/$val(bps)]
    $cbr_($i) set random_ 0
    $cbr_($i) attach-agent $udp_($i)
    $ns_ connect $udp_($i) $sink_($i)
    $ns_ at $i "$cbr_($i) start"
}
```

The scene movement file used, *drtest-2-1-scen*, was modified to provide an entire 350 m x 350 m movement field, which requires a 351 m x 351 m topography, since the actual 350 m boundaries are out-of-range in a 350 m x 350 m topography. This includes a continuous field from (0.0, 0.0) to (350.0, 350.0)[1] in the Cartesian plane. The scene movement file was configured to place `node_(0)` in the centre of a 350 m x 350 m field, at (175.0, 175.0), and `node_(1)` to start in the corner at (0.0, 0.0), at 247.5 m range, then at time 0.2, move through the centre (0.0 m range) to the opposite corner at (350.0, 350.0), again at 247.5 m range, at 8.8 m/s. The relevant settings in the scene movement file were:

```
# topo 351x351
# 2 nodes, 1 connection
# start in corner at 247.5m range
# at 0.2, move through centre (0.0m) to opposite corner (247.5m)
#
$node_(0) set X_ 175.000000000000
$node_(0) set Y_ 175.000000000000
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 0.000000000000
$node_(1) set Y_ 0.000000000000
$node_(1) set Z_ 0.000000000000
$ns_ at 0.200000000000 "$node_(1) setdest 350.000000000000 350.00000000
0000 8.800000000000"
```

Movement from location (0.0, 0.0) to location (350.0, 350.0), covers a total distance of $\sqrt{350^2 + 350^2} = 494.9747468$ m. At 8.8 m/s, this will require $494.9747468/8.8 = 56.24713032$ seconds to complete. Movement starts almost immediately at time 0.2 and thus completes at time 56.44713032. The early start ensures the initial results involve movement from almost maximum range. The track passing right over the receiver ensures the results include closing distance, minimum range and enlarging distance; and the last few seconds of the simulation ensure that data from near maximum range $\sqrt{175^2 + 175^2} = 247.4873734$ m are captured. Although the CBR generator starts at time 1.0, after the node is already moving, at this time the node has moved only $8.8 * 0.8 = 7.04$ m and is still $247.4873734 - 7.04 = 240.4473734$ m from the receiver.

[1]A 351 m x 351 m topography provides a field from (0.0, 0.0) up to, but not including, (351.0, 351.0).

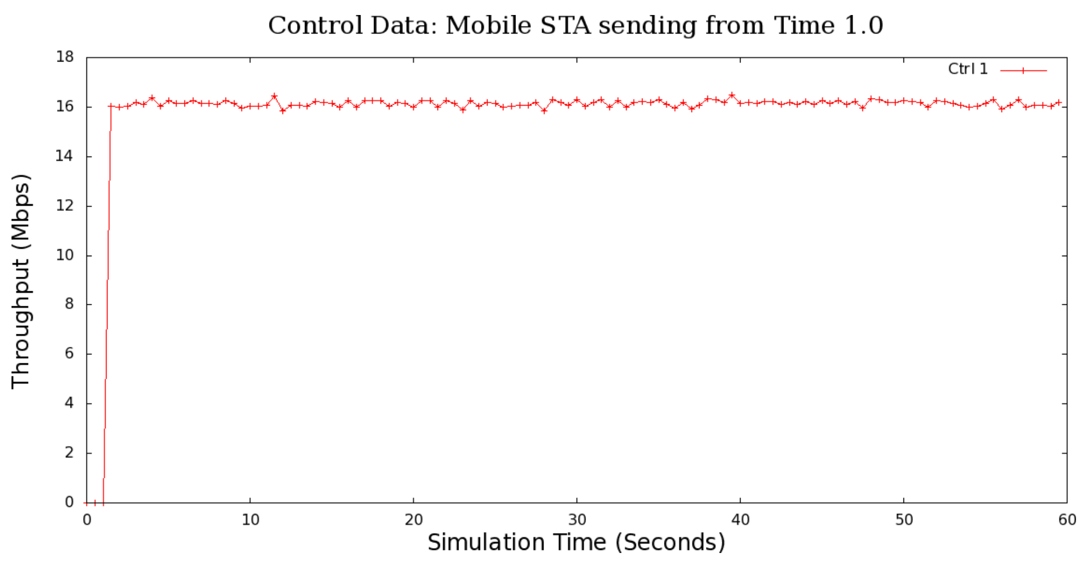


Figure 7.1: Control Data: Fully Mobile STA sending from Time 1.0

Figure 7.1 shows throughput for the chosen baseline control, with a "perfect" (noise free) 54 Mbps channel, transmitting packets at (an effective) 16.16 Mbps, with STA movement from near limit of range, through the receiver location and on to near limit of range again, starting at time 1.0 and settling to an average 16.15 Mbps thereafter, utilising 29.91% of the nominal channel bandwith.

The range delays included in this control have reduced the final average throughput from the fixed 1 m range results of Figure 6.18, with an average 16.16 Mbps throughput, utilising 29.93% of the nominal channel bandwith, to an average 16.15 Mbps throughput, utilising 29.91% of the nominal channel bandwith.

These data form the baselines for all following simulations. Additional control cases, involving multiple nodes in contention for access to the channel, were then prepared as follows.

7.1.1 A Note on Statistical Confidence

As the *ns-2* simulator is designed to be deterministic, so as to render completely reproducible results for any of the scenarios presented in this work, successive runs of any of these simulations will always produce exactly the same results, no matter how many times they are executed. As such, any confidence interval for any datum presented in this work will always be a point.

While repeating all of this research multiple times under multiple different seeds for the pseudo-random number generators (PRNG), in order to render a non-zero confidence interval is beyond the scope of this work and has been shown to be detrimental to the quality of the random numbers so produced [168] — we can use the fact that the *ns-2* PRNG functions have been demonstrated [168] to provide a uniform distribution, coupled with the fact that averaging a random sequence should always be the same for any random sequence to provide confidence intervals for the steady-state averages of data within any particular simulation, however as already stated, any confidence interval for an individual datum within a simulation is meaningless under these deterministic conditions.

Using the control data above, taking the steady-state data from time 3.0 onwards, the average throughput is 16,148,976 bps, with a standard deviation of 119,039 over 114 samples, gives a 95% CI of just ±21,852 or 0.135% of the mean or a a 99% CI of just ±28,718 or 0.178% of the mean. This level of confidence is typical of the results herein and such small intervals are of no consequence in determining the viability of this proposal.

7.1.2 Multi-Station Contention

Figure 7.2 shows the results of *drtesth.tcl* configured with two STAs (STA) in contention for the single 54 Mbps wireless channel, as follows:

```
set val(nn)       3                          ;# number of mobilenodes
set val(sc)       "drtest-3-2-scen-stat"     ;# scene movement file
set val(bps)      16.0                       ;# cbr Mbit rate
set val(stop)     60.0                       ;# simulation time

for {set i 1} {$i < $val(nn) } {incr i} {
    set udp_($i) [new Agent/UDP]
    $ns_ attach-agent $node_($i) $udp_($i)
    set sink_($i) [new Agent/LossMonitor]
    $ns_ attach-agent $node_(0) $sink_($i)
    set cbr_($i) [new Application/Traffic/CBR]
    $cbr_($i) set packetSize_ 2048
    $cbr_($i) set interval_ [expr 0.016384/$val(bps)]
    $cbr_($i) set random_ 0
    $cbr_($i) attach-agent $udp_($i)
    $ns_ connect $udp_($i) $sink_($i)
    $ns_ at $i "$cbr_($i) start"
}
```

Here, the scene movement file used, *drtest-3-2-scen-stat*, has two sending nodes 10 m either side of the central `node_(0)` receiver, for the entire simulation, with no movement and remaining within the 250 m radio range throughout the simulation. Thus, the receiver can "hear" both sending nodes and each node can hear both the receiver and the other node. The relevant settings in the scene movement file were:

```
$node_(0) set X_ 175.000000000000
$node_(0) set Y_ 175.000000000000
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 165.000000000000
$node_(1) set Y_ 175.000000000000
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 185.000000000000
$node_(2) set Y_ 175.000000000000
$node_(2) set Z_ 0.000000000000
```

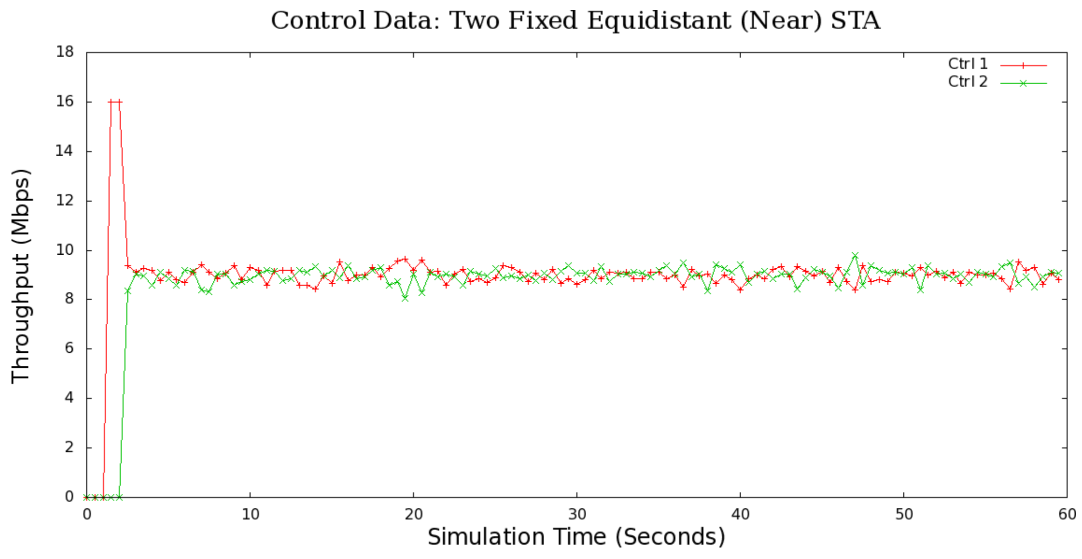


Figure 7.2: Control Data: Two Fixed Equidistant STA, all in range

Figure 7.2 shows the CBR generator of `node_(1)` starting at time 1.0 and quickly achieving an average throughput of 16.01 Mbps in the first second of operation. This is identical to the average over the first second of operation for the control data in Figure 7.1, at 16.01 Mbps, but due to the movement and differing ranges, this actually involves different data making up the average.

At time 2.0, the CBR generator of `node_(2)` starts and `node_(2)` contends with `node_(1)` for the channel. In the next second, `node_(1)` throughput drops to 9.248 Mbps and `node_(2)` throughput rises to 8.702 Mbps, achieving an average combined throughput of 17.95 Mbps. For the remainder of the simulation, `node_(1)` averages 9.000 Mbps throughput or 55.71% of our baseline rate and `node_(2)` averages 8.985 Mbps throughput or 55.61% of our baseline rate, with the average combined throughput being 17.98 Mbps (111.32% of the baseline), utilising 33.31% of the nominal channel bandwith.

This simulation was then repeated with the sending nodes 150 m each side of the `node_(0)` receiver, with no movement throughout the simulation. Thus, the receiver can hear both sending nodes and each node can hear the receiver but can not hear the other sending node. This is the so-called "hidden node" case. The relevant settings were configured in the scene movement file, *drtest-3-2-scen-dist*.

```
$node_(0) set X_ 175.000000000000
$node_(0) set Y_ 175.000000000000
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 25.000000000000
$node_(1) set Y_ 175.000000000000
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 325.000000000000
$node_(2) set Y_ 175.000000000000
$node_(2) set Z_ 0.000000000000
```

Figure 7.3 shows the CBR generator of `node_(1)` starting at time 1.0 and again achieving an average throughput of 16.01 Mbps in the first second of operation, identically to the close-range scenario and the control data in Figure 7.1.

At time 2.0, the CBR generator of `node_(2)` starts and although unable to hear `node_(1)`'s transmissions, has to contend with the flow of positive acknowledgements that it does receive from `node_(0)` for `node_(1)`, as well as wait for acknowledgements for its own transmissions to ensure any distant collisions are detected. In the next second, `node_(1)` throughput drops to 8.901 Mbps and `node_(2)` throughput rises to 9.331 Mbps, achieving an average combined throughput of 18.23 Mbps. For the remainder of the simulation, `node_(1)` aver-

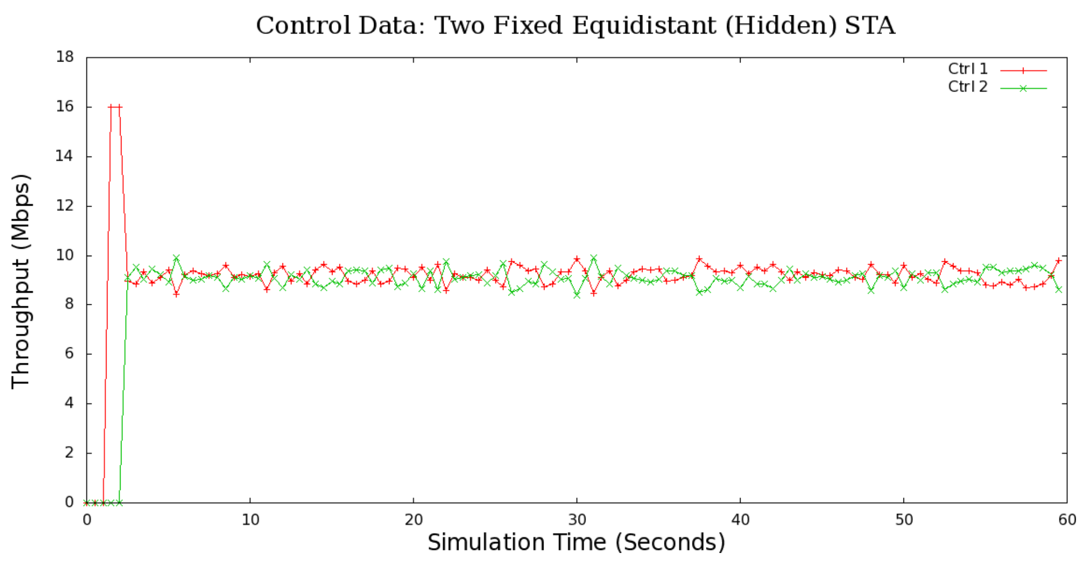


Figure 7.3: Control Data: Two Fixed Equidistant STA, hidden from each other

ages 9.213 Mbps throughput or 57.03% of our baseline rate and `node_(2)` averages 9.114 Mbps throughput or 56.41% of our baseline rate, with the average combined throughput being 18.33 Mbps (113.44% of the baseline), utilising 33.94% of the nominal channel bandwith.

This simulation yields slightly better results than the scenario where both sending STAs were able to hear each other's transmissions and indicates, for the *ns-2* simulator in this two STA and one AP case, that the impact of all three radios being in the same physical RF collision space is greater than the probability of having both outlying radios transmit at the same time.

The final case to be tested for the two-sending-node scenarios is where the nodes remain hidden from each other but are not equidistant from the receiver. This was configured with `node_(1)` 175 m on one side of the `node_(0)` receiver and `node_(2)` 90 m on the other side, with no movement throughout the simulation. Thus, the receiver can hear both sending nodes and each node can hear the receiver but can not hear the other sending node — hidden nodes — and each sender has a different propagation delay to the receiver. The relevant settings were configured in the scene movement file, *drtest-3-2-scen-diff*.

```
$node_(0) set X_ 175.000000000000
$node_(0) set Y_ 175.000000000000
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 0.000000000000
```

```
$node_(1) set Y_ 175.000000000000
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 265.000000000000
$node_(2) set Y_ 175.000000000000
$node_(2) set Z_ 0.000000000000
```

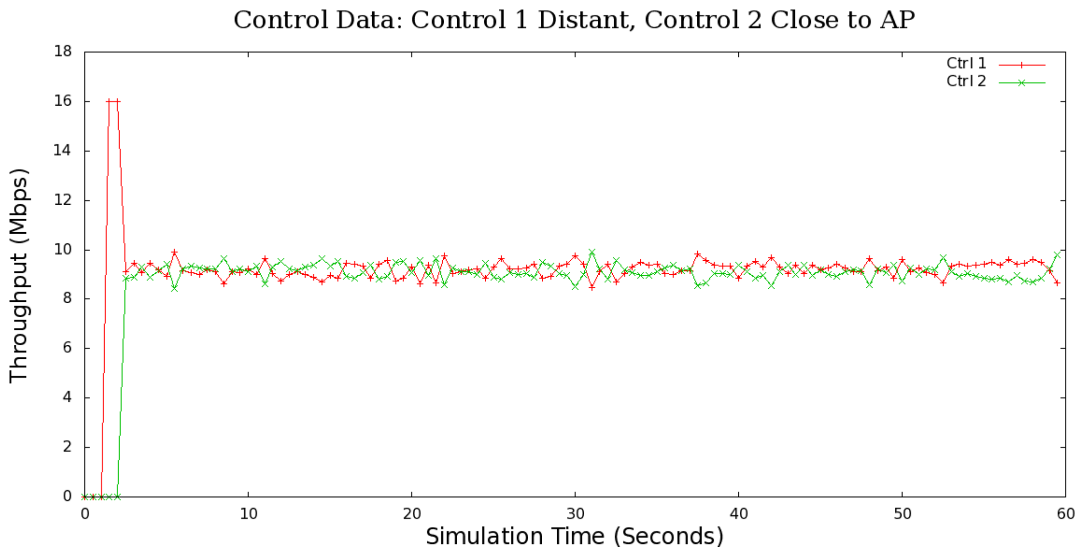


Figure 7.4: Control Data: Two Hidden STA, at different ranges

Figure 7.4 shows the CBR generator of `node_(1)` starting at time 1.0 and once again achieving an average throughput of 16.01 Mbps in the first second of operation, identically to the previous scenarios and the control data in Figure 7.1.

At time 2.0, the CBR generator of `node_(2)` starts and the two nodes now contend for the channel. In the next second, `node_(1)` throughput drops to 9.298 Mbps and `node_(2)` throughput rises to 8.884 Mbps, achieving an average combined throughput of 18.18 Mbps. For the remainder of the simulation, `node_(1)` averages 9.215 Mbps throughput or 57.04% of our baseline rate and `node_(2)` averages 9.115 Mbps throughput or 56.42% of our baseline rate, with the average combined throughput being 18.33 Mbps (113.45% of the baseline), utilising 33.94% of the nominal channel bandwith.

This simulation yields almost identical results to the previous scenario, except the differences in timing from the different distances of the two sending nodes have almost entirely inverted the two throughput traces in Figure 7.4, compared to Figure 7.3.

7.1.3 Recording Average Throughput

These results were combined in next simulation, which recorded the average throughput achieved by any one sending STA (out of two, in this case) in a single trace file. While this provides little advantage over calculating the average at each 0.5 second datum from the previous data, this tool becomes far more useful as the number of nodes increase and saves considerable time and effort in these cases. This was achieved by using the same scenario, including the same scene movement file, *drtest-3-2-scen-diff*, as the previous simulation and modifying the `proc record {}` to sum all the data received by all the sinks, divide by the number of active nodes and record the results in a single trace file `f0`, *drtestk-avgbw-2tnodes.tr*, as shown below.

```
proc record {} {
    global val f0 sink_ nodecount
    set ns_ [Simulator instance]
    set now [$ns_ now]
    set time 0.5
    set bz_ 0
    for {set i 1} {$i < $val(nn) } {incr i} {
        set by_ [$sink_($i) set bytes_]
        set bz_ [expr $bz_ + $by_]
        $sink_($i) set bytes_ 0
    }
    puts $f0 "[expr $now] [expr $bz_/$time*8/$nodecount]"
    $ns_ at [expr $now+$time] "record"
}
```

The results are shown in Figure 7.5.

Figure 7.5 shows the average throughput achieved by any one sending STA (out of two, in this case) in a single trace. This was performed using the same scenario, including the same scene movement file, *drtest-3-2-scen-diff*, as the previous simulation. These data were confirmed by calculating the average at each 0.5 second datum from the previous data in Figure 7.4 and subtracting the result from each matching datum here. In each and every case, the difference was zero, confirming correct operation of this averaging-simulation tool.

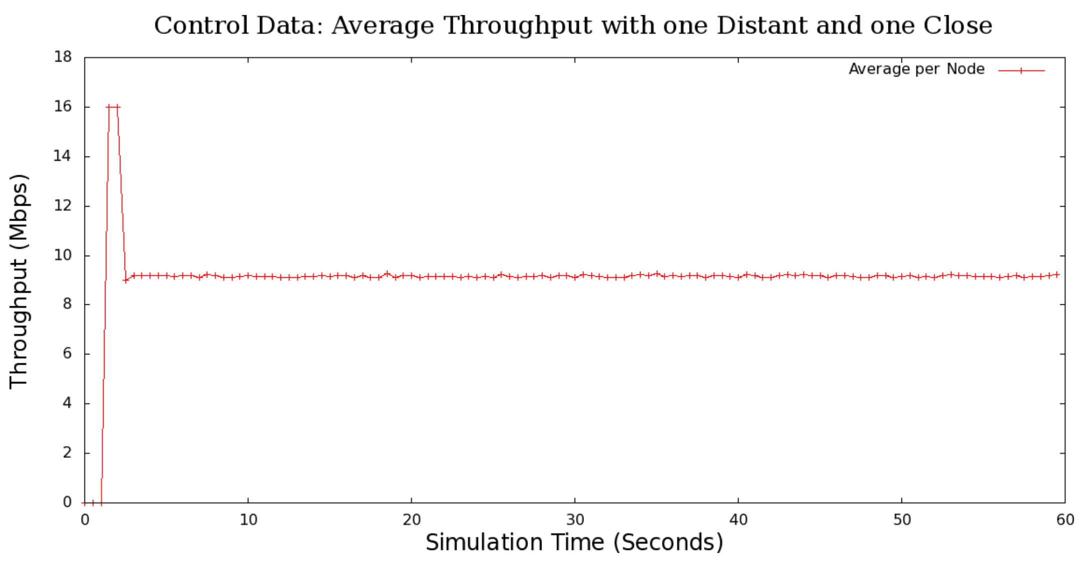


Figure 7.5: Control Data: Average Throughput per Node (Two Nodes)

7.1.4 Ten Stations with Extensive Contention

The next step was to develop control data for many high-output STAs in contention for the same channel. A random 11-node scene movement file, *drtest-11-10-scen*, was created with the following command.

```
./setdest -n 11 -p 2 -M 10.0 -t 60 -x 350 -y 350 >drtest-11-10-scen
```

This generates nodes in random positions with random movements, many of which would often be out of range of `node_(0)` at any given time, so this was then edited to change `node_(0)` to the centre of the topography at position (175.0, 175.0) and delete all movement commands for `node_(0)`. Changing this then renders all the General Operations Director (GOD) calculations in the scene movement file incorrect, so these too were removed, forcing *ns-2* to perform the calculations during the simulation run itself. This reduced the scene movement file to a list of initial positions followed by a series of node movements, within the 350 m x 350 m field, except for `node_(0)` fixed in the centre at (175.0, 175.0), as follows:

```
$node_(0) set X_ 175.000000000000
$node_(0) set Y_ 175.000000000000
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 24.390936066935
```

```
$node_(1) set Y_ 184.701008393152
...
$node_(10) set X_ 172.260937209305
$node_(10) set Y_ 40.659152666131
$node_(10) set Z_ 0.000000000000
$ns_ at 2.000000000000 "$node_(1) setdest 168.636731303167 159.39774932
2404 5.207680047769"
$ns_ at 2.000000000000 "$node_(2) setdest 92.127911570856 333.556068468
993 8.753310111512"
...
$ns_ at 2.000000000000 "$node_(10) setdest 251.212047386008 113.8202844
49872 1.543441734052"
$ns_ at 28.779308260520 "$node_(2) setdest 92.127911570856 333.55606846
8993 2.000000000000"
...
$ns_ at 50.101428659031 "$node_(3) setdest 90.940332993304 3.9273908333
61 4.850487587718"
$ns_ at 52.706904395870 "$node_(1) setdest 21.452592195044 301.19891812
8008 2.000000000000"
$ns_ at 54.706904395870 "$node_(1) setdest 297.162488370388 58.89371362
8615 0.296292012005"
```

This scene movement file was used in *drtestl.tcl*, which was configured to create 11 nodes, with CBR generators attached to UDP agents on `node_(1)` through `node_(10)`, each starting at time $5.0 * nodeindex$, to give five second intervals between increasing numbers of sending nodes; and all the sinks on `node_(0)`, with the throughput at each sink recorded every half-second, as shown below.

```
set val(nn)         11                          ;# number of mobilenodes
set val(sc)         "drtest-11-10-scen"         ;# scene movement file
set val(bps)        16.0                        ;# cbr Mbit rate
set val(stop)       60.0                        ;# simulation time

for {set i 1} {$i < $val(nn) } {incr i} {
    ...
    $ns_ attach-agent $node_($i) $udp_($i)
    $ns_ attach-agent $node_(0) $sink_($i)
    ...
    $cbr_($i) set packetSize_ 2048
    $cbr_($i) set interval_ [expr 0.016384/$val(bps)]
    ...
    $ns_ at [expr $i*5.0] "$cbr_($i) start"
}

proc record {} {
    ...
    for {set i 1} {$i < $val(nn) } {incr i} {
        set by_($i) [$sink_($i) set bytes_]
        puts $f_($i) "[expr $now] [expr $by_($i)/$time*8]"
        $sink_($i) set bytes_ 0
    }
    ...
}
```

The results are shown in Figure 7.6.

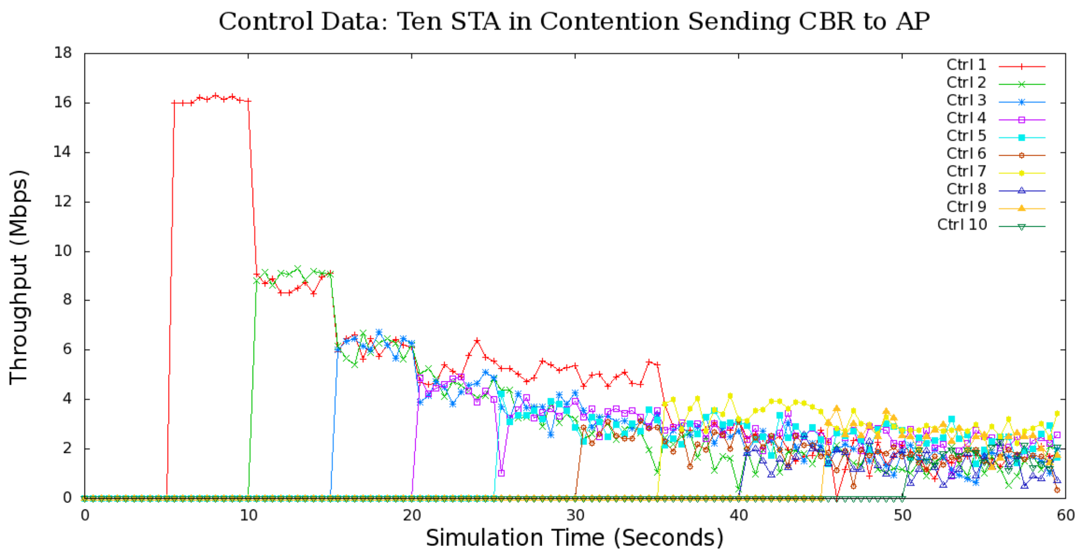


Figure 7.6: Control Data: Ten STA in Contention sending 16.16 Mbps each

Figure 7.6 shows the CBR generator of `node_(1)` starting at time 5.0 and initially achieving an average throughput of 16.01 Mbps in the first second of operation, identically to the previous scenarios and the control data in Figure 7.1, consolidating to a 16.13 Mbps average throughput over the first five seconds.

At time 10.0, the CBR generator of `node_(2)` starts and the two nodes now contend for the channel. In the next five seconds `node_(1)` throughput drops to 8.689 Mbps and `node_(2)` throughput rises to 9.033 Mbps, achieving an average combined throughput of 17.72 Mbps, equivalent to 8.861 Mbps per node.

At time 15.0, the CBR generator of `node_(3)` starts and the three nodes now all contend for the channel. In the next five seconds `node_(1)` and `node_(2)` throughput drops to 6.178 Mbps and 6.065 Mbps, respectively; and `node_(3)` throughput rises to 6.240 Mbps, making an average combined throughput of 18.48 Mbps, equivalent to 6.161 Mbps per node.

At time 20.0, the `node_(4)` starts and `node_(1)` through `node_(3)` throughput drops to 5.297 Mbps, 4.596 Mbps and 4.464 Mbps, respectively; and `node_(4)` throughput rises to 4.460 Mbps, giving an average combined throughput of 18.82 Mbps, or 4.704 Mbps per node.

At time 25.0, the `node_(5)` starts and `node_(1)` through `node_(4)` drop to 4.933 Mbps, 2.501 Mbps, 3.163 Mbps and 3.335 Mbps; and `node_(5)` rises to 3.511 Mbps, again slightly increasing the average combined throughput, now to 19.29 Mbps, or 3.859 Mbps per node.

At time 30.0, the `node_(6)` starts and the average combined throughput once again rises slightly, to 19.58 Mbps, or 3.263 Mbps per node.

At time 35.0, the `node_(7)` starts and now the average combined throughput drops slightly, to 19.06 Mbps, or 2.722 Mbps per node.

At time 40.0, the `node_(8)` starts and the average combined throughput drops again, to 18.92 Mbps, or 2.365 Mbps per node.

At time 45.0, the `node_(9)` starts and the average combined throughput picks up slightly, to 19.18 Mbps, or 2.131 Mbps per node.

At time 50.0, the `node_(10)` starts and the average combined throughput falls again, to 18.69 Mbps, or 1.869 Mbps per node.

These are summarised in the following Table 7.1.

For the remainder of the simulation, from time 51.0 to time 59.5 (18 samples), with all ten nodes active, the comparative throughput for each node and the whole system are summarised in Table 7.2.

Once again, the averaging-simulation tool discussed previously was applied to this scenario, recording the average throughput achieved by any one sending STA (out of ten, in this case) in a single trace file. This was achieved by using the same scenario, including the same scene movement file as the previous simulation,

Time	Active Nodes	Total Av. Throughput	Av. Throughput per Node
5.0	1	16,130,400	16,130,400
10.0	2	17,721,932.8	8,860,966.4
15.0	3	18,482,956.8	6,160,985.6
20.0	4	18,817,145.6	4,704,286.4
25.0	5	19,293,612.8	3,858,722.56
30.0	6	19,578,169.6	3,263,028.267
35.0	7	19,055,379.2	2,722,197.029
40.0	8	18,916,409.6	2,364,551.2
45.0	9	19,177,804.8	2,130,867.2
50.0	10	18,691,411.2	1,869,141.12

Table 7.1: Average Total Combined Throughput and Average Throughput per 16.16 Mbps Node

Node	Av. Throughput	% MAC Rate 54 Mbps	% Baseline 16.15625 Mbps
1	1,630,503.$\dot{1}$	3.019450206	10.09208889
2	1,433,813.$\dot{3}$	2.655209877	8.874666667
3	1,511,018.$\dot{6}$	2.798182716	9.352533333
4	2,229,763.$\dot{5}$	4.12919177	13.80124444
5	2,277,557.$\dot{3}$	4.217698765	14.09706667
6	1,558,812.$\dot{4}$	2.886689712	9.648355556
7	2,795,936.0	5.177659259	17.3056
8	1,246,314.$\dot{6}$	2.307990123	7.714133333
9	2,259,175.$\dot{1}$	4.183657613	13.98328889
10	1,720,576.0	3.186251852	10.6496
All	18,663,470.$\dot{2}$	34.56198189	115.5185778

Table 7.2: Average Throughputs for Ten Active 16.16 Mbps Nodes

and modifying the `proc record {}` to sum all the data received by all the sinks, divide by the number of active nodes and record the results in a single trace file. The results are shown in Figure 7.7.

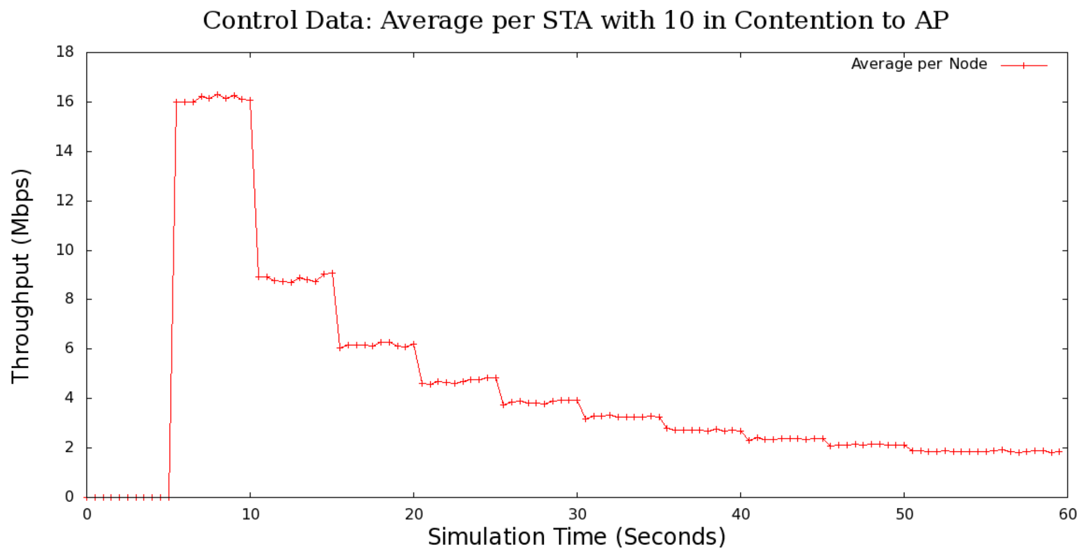


Figure 7.7: Control Data: Average per STA with 10 in Contention to AP

Figure 7.7 shows the average throughput achieved by any one sending STA in a single trace. This was performed using the same scenario, as the previous simulation. These data were confirmed by calculating the average at each 0.5 second datum from the previous data in Figure 7.6 and subtracting the result from each matching datum here. In each and every case, the difference was either exactly zero or a calculation rounding error 12 orders of magnitude smaller than the throughput, re-confirming correct operation of this averaging-simulation tool.

7.1.5 TCP Traffic

A baseline was established for TCP traffic. The TCP option of the CMU/Rice *cbrgen.tcl* tool was again used in an attempt to create a TCP traffic pattern file, *drtest-11-10-tcp-54*, using the following command.

```
ns cbrgen.tcl -type tcp -nn 10 -seed 1.0 -mc 10 -rate 0.0 >drtest-11-10
-tcp-54
```

Inspection of the resulting output revealed, for the same reasons as with the CBR generators, that this traffic pattern file was unsuitable for our purposes at this stage. Instead, the scenario was updated in *drtestn.tcl* to create and start FTP sources transferring a 2 MB file on TCP agents to and from alternate nodes and `node_(0)`, as shown below.

```
Agent/TCP set packetSize_ 2304

for {set i 1} {$i < $val(nn) } {incr i} {
    # ========================================TCP
    set tcp_($i) [new Agent/TCP]
    $ns_ attach-agent $node_($i) $tcp_($i)
    set sink_($i) [new Agent/TCPSink]
    $ns_ attach-agent $node_(0) $sink_($i)
    $ns_ connect $tcp_($i) $sink_($i)
    $tcp_($i) set window_ 32
    $tcp_($i) set packetSize_ 512
    set ftp_($i) [new Application/FTP]
    $ftp_($i) attach-agent $tcp_($i)
    $ns_ at [expr $i*5.0] "$ftp_($i) send 2000000"
    # ========================================TCP
# Bi-Directional
    incr i
    # ========================================TCP
    set tcp_($i) [new Agent/TCP]
    $ns_ attach-agent $node_(0) $tcp_($i)
    set sink_($i) [new Agent/TCPSink]
    $ns_ attach-agent $node_($i) $sink_($i)
    $ns_ connect $tcp_($i) $sink_($i)
    $tcp_($i) set window_ 32
    $tcp_($i) set packetSize_ 512
    set ftp_($i) [new Application/FTP]
    $ftp_($i) attach-agent $tcp_($i)
    $ns_ at [expr $i*5.0] "$ftp_($i) send 2000000"
    # ========================================TCP
}
```

Note that the TCP window size (set in packets in *ns-2*, not bytes) is only read on agent instantiation or reset, so the above settings, `set window_ 32`, have no effect and the default 20 packet window was applied. Conversely, the TCP packet size is used throughout the agent code an so the `set packetSize_ 512` overrides the global `set packetSize_ 2304` and the simulated packets contained 512 bytes of FTP, along with 60 bytes of TCP, IP and (extra) DSDV[2] IP.

The same 11-node scene movement file, *drtest-11-10-scen*, from the previous simulations was used. The results are shown in Figure 7.8.

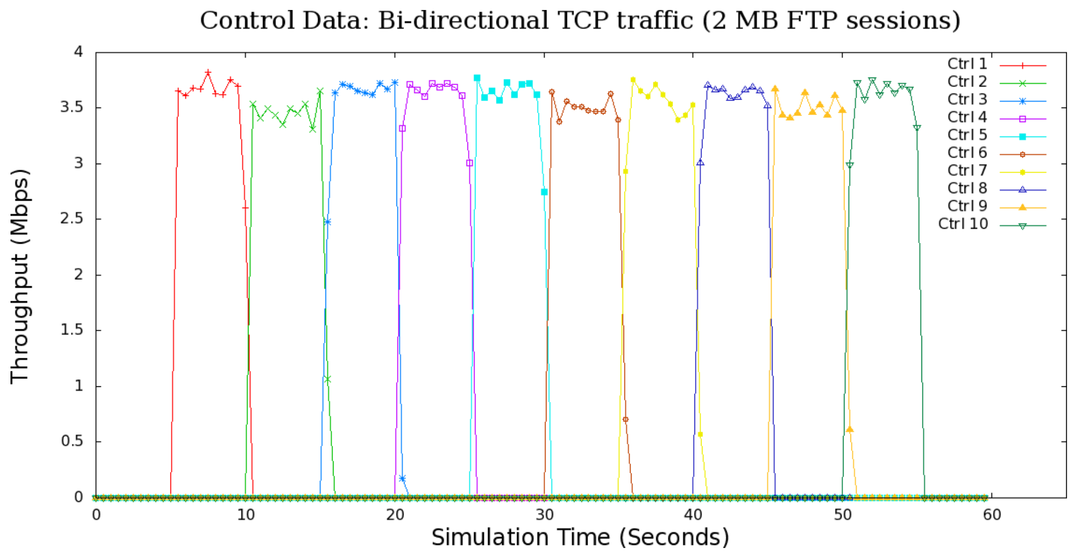


Figure 7.8: Control Data: Bi-directional 2 MB FTP sessions

Figure 7.8 shows the FTP generator of each of ten connections, `node_(1)` to `node_(0)`, `node_(0)` to `node_(2)`, `node_(3)` to `node_(0)`, `node_(0)` to `node_(4)`, `node_(5)` to `node_(0)`, `node_(0)` to `node_(6)`, `node_(7)` to `node_(0)`, `node_(0)` to `node_(8)`, `node_(9)` to `node_(0)` and `node_(0)` to `node_(10)`, starting at 5.0 second intervals and delivering a 2 MB FTP payload.

The first transfer completed before the second transfer started. However the second slightly overlapped the third, causing the third to very slightly overlap the fourth. The fourth completed before the fifth started, and the fifth before the sixth, but the sixth slightly overlapped the seventh, and the seventh also into the eighth. The eighth completed before the ninth, but the ninth overlapped the tenth slightly. The tenth finished by time 55.0.

In this case, all ten 2 MB FTP sessions completed within 50 seconds of the first start at time 5.0, transferring 2,234,864 bytes each — a 60-byte TCP SYN plus 3,907 x 572-byte TCP data packets including headers (512-byte FTP payloads) — within that time, at varying ranges with varying movement of nodes.

[2]Refer previous discussions about the *ns-2* DSDV routing functions adding extra headers.

The individual performance of each connection is summarised in the following Table 7.3. In each case, the average steady throughput is given, discounting overlapping or trailing time periods. The average of all ten of these steady throughputs was 3.611 Mbps (0.4514 MBps) and the overall average throughput, including trailing periods and between transfers, for the 50 seconds, was $\frac{2{,}234{,}864*10*8}{50.0} = 3,575,782.4$ bps or 3.576 Mbps or 0.4470 MBps.

Connection	Time Averaged	Av. Steady Throughput
1–0	5.0–9.5	3,683,278.$\dot{2}$
0–2	10.0–15.0	3,468,704.0
3–0	15.5–20.0	3,678,087.$\dot{1}$
0–4	20.5–24.5	3,677,960.0
5–0	25.0–29.5	3,668,024.$\dot{8}$
0–6	30.0–35.0	3,505,312.0
7–0	35.5–40.0	3,584,533.$\dot{3}$
0–8	40.5–44.5	3,652,792.0
9–0	45.0–50.0	3,514,464.0
0–10	50.5–54.5	3,679,104.0
All	Average of the above	3,611,225.9$\dot{5}$
All	Overall (5.0–55.0)	3,575,782.4

Table 7.3: Average TCP Throughput per node

This simulation was also executed recording the total throughput of all nodes in a single trace. The results are shown in Figure 7.9.

Figure 7.9 shows the total throughput at any given time for all sending STAs in a single trace. This was performed using the same scenario as the previous simulation and gives the overall average throughput, from time 5.5 to time 55.0, as 3,575,782.4 bps or 3.576 Mbps, identical to the previously calculated results.

7.1.6 Overlapping TCP Traffic

The previous scenario was updated in *drtestp.tcl* to use FTP sources transferring a 2.6 MB files, instead of the previous 2.0 MB files, as shown below.

```
set val(stop)      80.0      ;# simulation time extended
...
    $ns_ at [expr $i*5.0] "$ftp_($i) send 2600000"
...
```

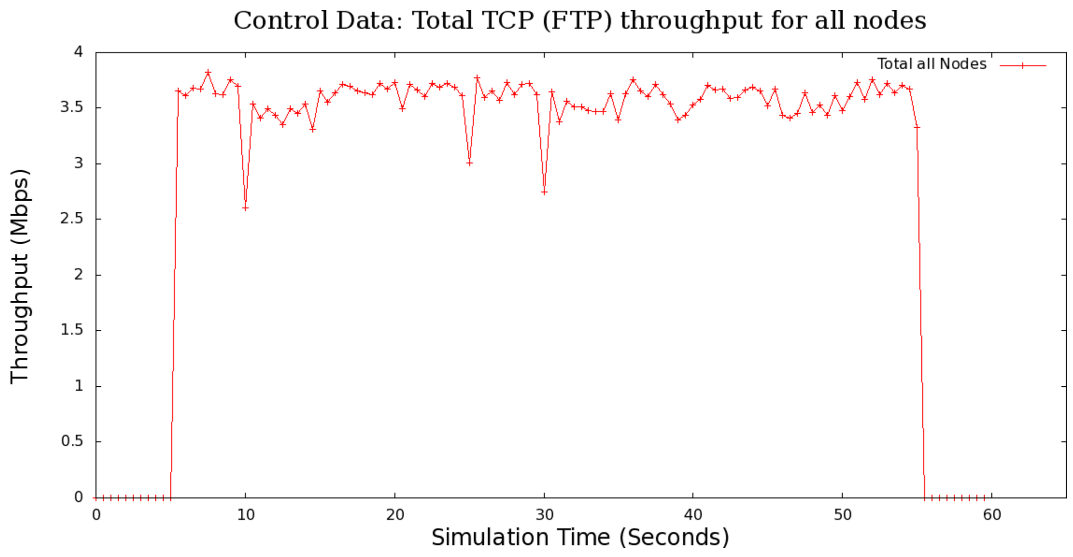


Figure 7.9: Control Data: Total TCP (FTP) throughput for all nodes

The same 11-node scene movement file with `node_(0)` stationary in the centre and the other ten nodes moving, *drtest-11-10-scen*, from the previous simulations was used. The results are shown in Figure 7.10.

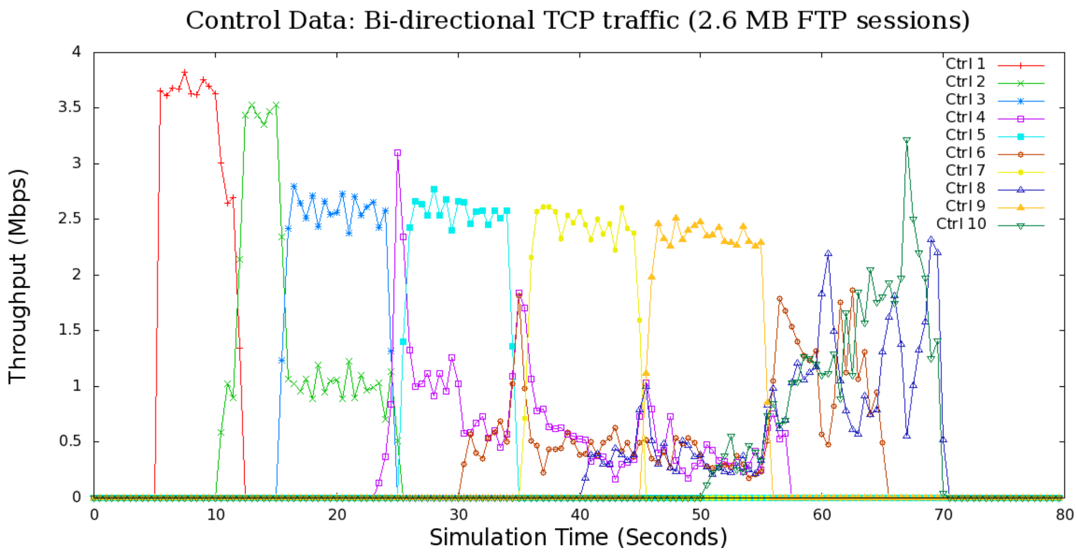


Figure 7.10: Control Data: Bi-directional Overlapping 2.6 MB FTP sessions

Figure 7.10 shows the FTP generator of each of ten connections, 1–0, 0–2, 3–0, 0–4, 5–0, 0–6, 7–0, 0–8, 9–0 and 0–10, starting at 5.0 second intervals and delivering a 2.6 MB payload.

At the start of the simulation, the channel is quiet. At 0.089076526, the `node_(3)` DSDV routing function broadcasts a 32-byte routing update, consisting of a 12-byte[3] DSDV route advertisement and a 20-byte IP header, to the LLC layer for IP broadcast. In *ns-2*, the LLC does not add the 8-byte LLC header to the packet, neither in size nor content. The LLC sets the MAC broadcast address (which gets added to the packet in the MAC layer) and sends the 32-byte packet to the MAC layer. The MAC layer adds `phymib_.getHdrLen11()` ((`PreambleLength` + `PLCPHeaderLength`) / 8 + MAC header length + `ETHER_FCS_LEN` = (72 + 48) / 8 + (2+2+6+6+6+6+2)[4] + 4 = 15 + 30 + 4 = 49 bytes) to the packet size, making the packet size 81 bytes for transmission. At 0.089171526, 95 µs later, the `node_(3)` MAC layer broadcasts it to all STA. This 95 µs delay consists of the 25 µs LLC delay, a 50 µs DIFS (10 µs SIFS + 2 x 20 µs slot times) and one 20 µs slot time.

Only nodes 1, 6, 4, 0, 2, 7 and 5 are in range to receive the message. From 0.089819879 to 0.089820358, the MAC layers of nodes 1, 6, 4, 0, 2, 7 and 5 receive the message, taking, in seconds,
0.000648353,
0.000648473,
0.000648494,
0.000648509,
0.000648631,
0.000648774, and
0.000648832, respectively. These times are made up of the transmission time for the 81-byte broadcast packet, which is always sent at the applicable basic rate ((81 * 8 = 648 bits) / 1 Mbps = 648 µs), plus the propagation time to the node (distance / `SPEED_OF_LIGHT` = 0–250 / 300,000,000 = 0–833 ns).

The receiving MAC layers then pass it to their routing functions for processing with a 25 µs link layer delay.

At 0.192938868, the `node_(9)` routing function sends its own advertisement for broadcast. In this case, 655 µs elapses before the MAC layer broadcasts the 81-byte MPDU. The 655 µs delay consists of the 25 µs LLC delay, a 50 µs DIFS and *twenty-nine* 20 µs slot times, drawn randomly from the current 31-slot

[3]The current DSDV implementation in ns-2 actually stores route advertisements in 9 bytes (4-byte destination, 1-byte metric and 4-byte serial) but sets the packet size as if advertisements are *12 bytes* each, using change_count * 12 + IP_HDR_LEN.

[4]*ns-2* structures the MAC header as FC:Du:RA:TA:3A:4A:SC, as opposed to the actual FC:Du:RA:TA:3A:SC:4A.

Contention Window (CW). The minimum, CWMin_, is 31 slots (0–31 slot delay) and the maximum, as a result of backoff, CWMax_, is 1023 slots (0–1023 slot delay).

Here, only nodes 5, 4, 2, 8, 1, 10, 7 and 0 are in range to receive the message. From 0.194241973, the MAC layers of nodes 5, 4, 2, 8, 1, 10, 7 and 0 receive the message and pass it to their routing functions for processing, again with a 25 µs link layer delay.

At 0.229788092, the node_(5) routing function does the same. In this case, 115 µs elapses (LLC + DIFS + 2 slot times) before the node_(5) MAC layer broadcasts it to the nodes in range, 9, 2, 4, 8, 10, 1, 7, 0 and 3.

At 0.335913132, the node_(2) routing function repeats this process. This time, 155 µs elapses (LLC + DIFS + 4 slot times) before the MAC layer broadcasts it to nodes 10, 7, 0, 8, 5, 4, 1, 9, 3 and 6 (all nodes are in range of node_(2) at this time).

At 0.353652583, node_(8)'s routing function advertises and 595 µs elapses (LLC + DIFS + 26 slot times) before the MAC layer broadcasts it; and at 0.627071594, node_(0)'s routing sends, with 135 µs (LLC + DIFS + 3 slot times) before the MAC layer uses the channel. node_(7) takes its turn at 0.853263746, with a 355 µs (LLC + DIFS + 14 slot times) delay; node_(4) at 1.282705256, with a 75 µs (LLC + DIFS + 0 slot times) delay; node_(10) at 1.616675349, with a 295 µs (LLC + DIFS + 11 slot times) delay; node_(1) at 1.675920244, with 315 µs (LLC + DIFS + 12 slot times); and node_(6) at 1.727182352, with 555 µs (LLC + DIFS + 24 slot times) elapsing before the MAC layer uses the channel. The channel then quiesces again until the first of the FTP transfers.

The node_(1) starts at position (24.390936066935, 184.701008393152) and at time 2.0 begins to move towards (168.636731303167, 159.397749322404) at 5.207680047769 ms^{-1}.

The first transfer starts at time 5.0 with exclusive use of the channel, achieving an average 3.683 Mbps from time 5.5 to time 9.5, identically to Figure 7.8.

At 5.000000000, the TCP agent on node_(1) sends a 40-byte[5] TCP SYN for node_(0) to the routing functions. DSDV adds another 20-byte IP header and passes it to the LLC layer. The LLC cannot resolve the address and the ARP functions hold the packet and generate an ARP request, passing it to the MAC

[5]The *ns-2* TCP agent already includes both the TCP and IP headers in the packet size. Both the DSDV and AODV wireless ad hoc routing elements also add another another 20-byte IP header to the packet size. The DSDV routing also adds yet another 20-byte IP header every time a packet is re-queued.

layer after the LLC processing delay.

At time 5.000025, `node_(1)` should have moved 15.62317034 m to position (39.77914228, 182.0016454), 135.4020066 m from `node_(0)` at (175.0, 175.0). The MAC layer adds 49 bytes of PLCP and MAC headers to the 28-byte ARP MSDU and *ns-2* calculates transmission time, distances and propagation times at this stage.

After a further 150 µs (DIFS + 5 slot times), the MAC broadcasts the 77-byte ARP request to all STA. The broadcast MPDU's are sent at 1 Mbps, taking 616 µs ((77 * 8 = 616 bits) / 1 Mbps) to transmit and have a propagation delay of 0–833 ns, as shown previously.

The receiving MAC layers strip off the 49 bytes of PLCP and MAC headers and pass the 28-byte ARP MSDU to their respective LLC without delay. The LLC layers immediately pass the ARP requests to their ARP modules. At time 5.000791451, the `node_(0)` ARP module processes the ARP request and returns a unicast ARP reply to its MAC layer after an LLC processing delay.

At time 5.000816451, after the 25 µs for the LLC, `node_(1)` should have moved a total of 15.62729196 m — a difference of little more than 4.12 mm — to be only about 4.09 mm closer to `node_(0)`. The MAC layer adds 49 bytes of PLCP and MAC headers to the MSDU and, 95 µs (LLC + DIFS + 1 slot time) after receiving the ARP request, transmits the reply packet.

Although the ARP request was a broadcast MPDU, sent at 1 Mbps, the ARP reply is unicast and sent at the Data Rate of 54 Mbps. However, even for a unicast packet, the 144-bit (long) or 72-bit (short) preamble (128-bit or 56-bit sync and 16-bit SFD) is still always transmitted at 1 Mbps and the 48-bit PLCP header is transmitted at 1 Mbps (long) or 2 Mbps (short). In *ns-2*, transmission time is calculated using only a single rate for the entire PLCP preamble and header, the `PLCPDataRate_`, which is set to '`6.0e6`' or 1 Mbps; and then either the `basicRate_` (also 1 Mbps) is used for the remainder of management, control or broadcast frames; or the `dataRate_` (54 Mbps) is used for the remaining bits of data frames.

The (broadcast) ARP request took 616 µs to transmit and, with the total delay recorded in the *ns-2* trace file being 0.000616451 s, demonstrated a propagation delay of 451 ns. This agrees with the 451.3400221 ns calculated from the extrapolation of the range from initial positions (135.4020066 m / 3x10^{8} ms^{-1}). The ARP reply took only $129.\dot{1}8\dot{5}$ µs (120 bits / 1 Mbps + 496 bits / 54 Mbps) to transmit and, with the total recorded delay being 0.000129637 s, demonstrated a propagation delay of 451.8148145 ns, within half a nanosecond[6] of the 451.3263853 ns calculated from the extrapolation of the range from initial positions (135.3979156 m / 3x10^{8} ms^{-1}). Note that the trace file records no digits past whole nanoseconds here.

After waiting a 10 µs SIFS, the `node_(1)` MAC layer sends an 802.11 MAC-layer ACK to `node_(0)`, taking 232 µs to transmit. Meanwhile `node_(1)` updates its ARP tables and sends the waiting TCP SYN to the MAC layer after 25 µs of LLC processing. As the medium is not idle, the *ns-2* MAC "defers" transmission by using the its "back-off" timer[7] in a paused state, waiting for the medium to become idle.

The MAC-layer ACK takes 232 µs to transmit and a further 451.326 ns for the last bit to reach `node_(0)`. This gives a total round-trip time from `node_(0)` starting to transmit the ARP reply of $129.\dot{1}8\dot{5}$ µs + 451.340 ns + 10 µs + 232 µs + 451.326 ns = 372.088 µs, well within the `txtime(pktTx_)` + `DSSS_MaxPropagationDelay` + `getSIFS()` + `txtime(getACKlen(), basicRate_)` + `DSSS_MaxPropagationDelay` = $129.\dot{1}8\dot{5}$ µs + 2 µs + 10 µs + 232 µs + 2 µs = $375.\dot{1}8\dot{5}$ µs time-out before a retransmit is scheduled.

After the MAC-layer ACK has been transmitted, the `node_(1)` MAC layer then waits a further 310 µs (DIFS + 13 slot times), then sends the waiting SYN to `node_(0)`.

On receiving the TCP SYN, `node_(0)` waits a SIFS and then sends a MAC-layer ACK to `node_(1)`.

It takes 25 µs through LLC for the SYN to get to the `node_(0)` transport layer. The TCP agent generates the transport-layer SYN-ACK and passes it to the routing functions, which in turn pass the routed segment to the LLC, which hands it on to the MAC layer after a further 25 µs. However, once again, the node is still transmitting the MAC-layer ACK and the MAC waits for the transmission to finish — a SIFS plus 232 µs after the TCP SYN was received — then defers a further DIFS plus 11 slot times, before transmitting the transport-layer SYN-ACK.

[6]The 0.4884292 ns difference represents the 11th-least-significant digit in the calculation, demonstrating a discrepancy of 0.0000000004884292 s in 5.000886451 s or a fraction of 9.77x10^{-11}, most likely due to rounding the least significant digit recorded in the trace file.

[7]This is a "defer" operation, but the back-off timer is used because it pauses while the medium is busy, whereas the defer timer in *ns-2* does not. By default, the defer timer in *ns-2* is now only used to wait a SIFS before an ACK or CTS.

Also during this time, `node_(1)` finishes receiving the MAC-layer ACK before its retransmit time-out expires.

On receiving the TCP SYN-ACK, `node_(1)` sends a MAC-layer ACK to `node_(0)`, after waiting a 10 µs SIFS. After 25 µs (LLC) the SYN-ACK is received at the transport layer. The TCP agent piggybacks the transport-layer ACK to the first 512-byte data payload, packet 13 in this simulation[8], with TCP sequence 1, along with 40-bytes of TCP and IP headers and passes it to the routing functions, which add an extra 20-byte IP header and pass the routed segment to the LLC, while the TCP agent prepares a second 552-byte segment, packet 14, TCP sequence 2, and also passes this one on to the routing functions, which add the extra 20-byte IP header and pass it on to the LLC. There are now queued packets at the LLC.

The (now 572-byte) packet 13 is handed on to the MAC layer after a further 25 µs in the LLC sub-layer. Another 49 bytes of MAC and PLCP headers are added to the packet, making it 621 bytes in total. Again, the node is still transmitting the MAC-layer ACK and the MAC waits for the transmission to finish and then defers a further DIFS plus 10 slot times, before transmitting the 621-byte packet. In *ns-2*, the 120 bits of preamble and PLCP takes 120 µs to transmit, while the remaining 4,848 bits take only $89.\dot{7}$ µs to transmit.

Along with the current 451 ns propagation delay, `node_(0)` finishes receiving the packet 0.000210229 s later, waits a SIFS and sends the return MAC-layer ACK to `node_(1)`. After the LLC delay, the `node_(0)` TCP receives packet 13 and sends an empty transport-layer TCP ACK, packet 15, for TCP sequence 1, back to the LLC. After another LLC delay, the TCP ACK is received by the MAC layer, which waits for the MAC ACK transmission to finish and then defers a further DIFS plus 4 slot times, before transmitting the TCP ACK.

When the empty TCP ACK, packet 15, is received by `node_(1)`, the node waits a SIFS and returns a MAC ACK to `node_(0)`. The TCP agent receives the sequence 1 ACK and prepares the sequence 3 segment as packet 16 and sequence 4 as packet 17, passing them to the routing functions. Meanwhile, after sending the MAC ACK to `node_(0)`, `node_(1)` waits a further DIFS and 10 slot times and sends the queued packet 14 (TCP data seq 2) to `node_(0)`.

This pattern continues for some time. Every packet received by a node, causes it to wait a SIFS and then send a return MAC ACK. This takes 242 µs

[8]The MAC-layer ACKs are not assigned packet numbers in *ns-2*, however all route, discovery and other control or management packets are.

to complete. Meanwhile, the node processes the received packet and has another transport packet ready to send within 50 µs, but must wait while the MAC ACK is being transmitted.

Every time `node_(0)` receives a TCP data segment, it queues an empty TCP ACK within 50 µs, but must wait another 192 µs for the MAC ACK to finish transmitting, plus a 50 µs DIFS, plus a random number of 20 µs slot times from the contention window, before it can begin to transmit the TCP ACK, which then takes a further $133.\dot{9}2\dot{5}$ µs in this implementation, plus approximately 451 ns to propagate in this instance. Thus, every TCP data packet takes between 426.377 µs and 1,046.377 µs to acknowledge in perfect conditions, with no collisions.

For every TCP ACK received by `node_(1)`, at this stage, the TCP agent expands the TCP window and prepares two more TCP data packets. However the queued data packets must also wait 242 µs after every packet received for the MAC ACK to finish transmitting, plus a 50 µs DIFS, plus a random number of 20 µs slot times from the contention window, before it can begin to transmit the next TCP packet, which then takes a further $209.\dot{7}$ µs plus approximately 451 ns to propagate, in this case. This gives between 502.229 µs and 1,122.229 µs to send a queued TCP data packet in perfect conditions, with no collisions. Thus, the round-trip time to send and acknowledge a queued TCP data packet in this implementation is between 928.606 µs and 2,168.606 µs. The 572-byte data segment itself takes only $84.\dot{7}4\dot{0}$ µs to transmit.

Thus, the theoretical average throughput (MSDUs) in these circumstances is $84.\dot{7}4\dot{0}$ / 1548.606 = 5.472% or 4,576 bits / 1548.606 µs = 2.955 Mbps. However, our simulations give better results, above 3.615 Mbps to 3.826 Mbps.

As this exchange continues, sometimes the contention backoff selected by `node_(0)` is longer than the MAC ACK propagation time and the contention backoff selected by `node_(1)`. In this case, `node_(1)` transmits a second queued packet before `node_(0)` can send the TCP ACK. This also results in queued TCP ACK packets at `node_(0)`. Conversely, sometimes `node_(0)` will transmit a second queued TCP ACK before `node_(1)` can send another queued TCP data packet.

When `node_(1)` sends two data packets in series, it takes 242 µs after the previous packet received for the MAC ACK to finish transmitting, plus a 50 µs DIFS, plus a random number of 20 µs slot times from the contention window, plus $209.\dot{7}$ µs to transmit the first TCP packet, plus approximately 451 ns to propagate, plus a 10 µs SIFS, plus 232.451 µs to return the MAC ACK, plus a 50 µs DIFS, plus a random number of 20 µs slot times from the contention window, plus $209.\dot{7}$ µs to transmit the second TCP packet, which then takes a further 451 ns to propagate, in this case. This gives between 1,004.909 µs and 2,244.909 µs to send two queued TCP data packets with no collisions.

Similarly, when `node_(0)` sends two TCP ACK packets in series, it takes 242 µs after the previous packet received for the MAC ACK to finish transmitting, plus a 50 µs DIFS, plus a random number of 20 µs slot times from the contention window, plus $133.\dot{9}2\dot{5}$ µs to transmit the first TCP ACK, plus approximately 451 ns to propagate, plus 242.451 µs to return the MAC ACK, plus a 50 µs DIFS, plus a random number of 20 µs slot times from the contention window, plus $133.\dot{9}2\dot{5}$ µs to transmit the second TCP ACK, which then takes a further 451 ns to propagate. This gives between 853.205 µs and 2,093.205 µs to send two queued TCP ACK packets, again with no collisions.

The round-trip time to send and acknowledge two queued TCP data packets in this case is between 1,858.113 µs and 4,338.113 µs, or an average 1,549.057 µs per 572-byte segment. This produces a throughput of 2.954 Mbps, almost identical to the single-data-ACK exchange, marginally reduced due to the extra two small propagation delays.

Occasionally, a node may be able to transmit three or more packets, before the other selects a shorter slot position — but again these combinations make little difference to the throughput and only serve to reduce it slightly due to the node waiting for the extra propagation delays in receiving the MAC ACKs (ACK+tcp—ACK—tcp—ACK+ack—ACK—ack) — as opposed to transmitting a MAC ACK (one less propagation delay) in single-data-packet exchanges (ACK+tcp—ACK+ack—ACK+tcp—ACK+ack).

In all possible combinations, our simulations give better results, 3.615 Mbps to 3.826 Mbps, compared to the theoretical values, at best 2.955 Mbps.

During these calculations, it was noticed the slot times were rarely over half of the contention window. Only 44 of the first 256 packets had slot times greater than 15 slots. The average number of slot times for the first 256 packets was just 8.105 slots, from a 31-slot contention window.

An average 8.105 slots produces a round-trip time to send and acknowledge a queued TCP data packet of 928.606 µs plus 8.105 * 20 µs = 1,090.706 µs and a throughput of $84.\dot{7}4\dot{0}$ / 1090.706 = 7.769% or 4,576 bits / 1,090.706 µs = 4.195 Mbps. This is well above the simulation results, but may support some skewing of the results if this initial trend were not to become completely uniform over time.

While this is not a sufficiently large sample to interpolate the actual average slot time used, it demonstrated that the pseudo random number generator, or its use in these functions, was a likely candidate to affect the simulated throughput. The MAC Backoff Timer was investigated more closely.

The *ns-2* MAC Backoff Timer uses `Random::random() % cw`. Using modulo functions to derive random numbers often attracts criticism, since the modulo approach only uses the low order bits from the PRNG and may not be as uniform as using the high order bits in some implementations. This is not an issue for modern Linux implementations, as provided in the Linux manual:

> The versions of rand() and srand() in the Linux C Library use the same random number generator as random() and srandom(), so the lower-order bits should be as random as the higher-order bits. However, on older rand() implementations, and on current implementations on different systems, the lower-order bits are much less random than the higher-order bits. Do not use this function in applications intended to be portable when good randomness is needed. [169]

This reference also recommends using floating point arithmetic, of the form `(int)(cw * (rand() / (RAND_MAX + 1.0)))`, in order to utilise the high order bits.

This is not the problem in *ns-2*, as the simulator uses its own PRNGs from the `Random` class in the *~ns2/tools/rng.h* and *~ns2/tools/rng.cc* files, which do claim to produce a uniform distribution. The problem was indicated, not so much by the very low initial average, as by the low initial range. The slot times used for the first 256 packets, ranged from 0 to 29 from a 31-slot contention window.

Confusion over the DCF Backoff Interval

The IEEE 802.11 DCF backoff is a pseudorandom integral number of slots drawn from a uniform distribution over the interval `[0,CW]`.

However, numerous peer-reviewed academic works [170–173] on the DCF and various alternate backoff schemes state the pseudorandom number of slots is drawn from a uniform distribution over the interval `[0,CW-1]`, or often "`[0,w-1]`, where `w` is the contention window" [170, 171], or similar descriptions to that effect, but still cite the minimum and maximum contention window sizes as per the standard, appropriate for an interval of `[0,CW]`.

This error is being perpetuated throughout the current literature, apparently as authors rely on this existing literature without referring to the standards.

Every version of the IEEE Std 802.11 known to this author [5,9,122,174,175] states that the `Backoff Time` consists of a "pseudorandom integer drawn from a uniform distribution over the interval `[0,CW]`" [5] times the `SlotTime` value. The interval is inclusive as indicated by the closed square brackets — using neither a parenthesis, '`[0,CW)`', nor the ISO notation of an outwards pointing bracket, '`[0,CW[`', to indicate exclusion of the endpoint — and provides the minimum contention window sizes as 15, 31 and 63 slots for each of the FHSS, DSSS and IR PHY, respectively and the maximum contention window size as 1023 slots for all three.

However, the *ns-2* MAC Backoff Timer's `Random::random() % cw` only provides the interval `[0,CW-1]`, for these same window sizes — the minimum 31 slots in the simulations here. This is an error in *ns-2* and the function should at least be `Random::random() % (cw + 1)` to provide the complete interval `[0,CW]`.

As stated above, this error also appears frequently in the current literature. In general, these works [170–173, 176] reference the IEEE Std 802.11, but most, notably, have also utilised the *ns-2* simulator. However the *ns-2* implementation does not appear to be the source of this misdirection, but simply another symptom of the underlying issue.

In general, the older works from around 1997, almost always use the interval "`[0,w-1]`", where "`w` is the contention window", in particular Bianchi [177] states this and then offers the "values `CWmin` and `CWmax` reported in the final version of the standard [cites a 1997 version] are PHY-specific and are summarised in Table I" [177, p. 536], which gives the various `CWmin` as 16, 32, and 64, for FHSS, DSSS and IR, respectively and `CWmax` as 1024 for each.

The recorded 1997 version of the standard [5] currently available from IEEE, states the random integer for DCF backoff is drawn from "a uniform distribution over the interval `[0,CW]`[9], where `CW` is an integer within the range of values of the MIB attributes `aCWmin` and `aCWmax`, `aCWmin`$\leq$`CW`$\leq$`aCWmax`" [5, p. 76] and provides

[9]Note that this is for DCF backoff — the PCF uses the range from 1 to aCWmin.

`aCWmin` as 15, 31 and 63, for FHSS, DSSS and IR, respectively and `aCWmax` as 1023 for each [5, pp. 218, 223, 237 and 272]. Clearly both descriptions produce the same intervals and it would appear that what Bianchi refers to pre-dates the actual 1997 version of the standard as it is now recorded.

This transition is well covered by Wu [178], who uses "a uniform distribution over the interval `[0,CW]`", but also describes:

> "The back-off time is uniformly chosen in the range `(0, w-1)`. Also `(w-1)` is known as Contention Window(CW), which is an integer with the range determined by the PHY characteristics `CWmin` and `CWmax`."

It is this fact that `CW` replaces `w-1`, with the standardised minimums of 15, 31 and 63, instead of 16, 32 and 64, that has been omitted by many authors.

Thus, new works relying on these older works may contain a mixture of pre- and post-standard descriptions, such as [171] which uses `[0,w-1]` on the first page and `[0,CW]` on the second page — in both cases referring to the standard IEEE 802.11 DCF. This should not be confused with other works, such as [179], describing new paradigms, who as a matter of design, use both `[0,CWi]` and `[0,CWi-1]` in their proposed algorithms, depending on whether they are increasing or decreasing their current window.

While works, such as [177,180], using the `[0,w-1]`, with `CWmin` and `CWmax` as powers of 2, still produce the correct intervals, others, such as [170–173, 176], using this interval with the standard `CWmin` and `CWmax` as powers of 2 minus 1 are therefore incorrect.

The ns-2 DCF Backoff Interval

The *ns-2* implementation falls into this category, drawing a random number backoff slots from the incorrect interval `[0,CW-1]`, combined with the standardised values `CWmin_ = 31` and `CWmax_ = 1023`.

Even with a uniform distribution, this reduced interval reduces the average number of slot times to 15, giving 928.606 µs plus 15 * 20 µs = 1,228.606 µs and a best-case throughput (single-data-ack exchanges) of $84.\dot{7}4\dot{0}$ / 1228.606 = 6.897% or 4,576 bits / 1,228.606 µs = 3.7245 Mbps, as supported by our simulation results.

In this case, where the node is now transferring a larger FTP payload, the average over the first 5 seconds, from time 5.5 to 10.0 is now 3.678 Mbps, which is more than the previous 3.576 Mbps, where the FTP session completed before time 10.0.

At time 10.0, the second transfer, outbound from `node_(0)` to `node_(2)`, starts, but must contend with the transfer still being received by `node_(0)` from `node_(1)`. The TCP agent on `node_(0)` sends a SYN (packet 8060) for `node_(2)` at precisely 10.000000000 and the DSDV routing protocol adds its 20-byte header and passes it to the LLC sub-layer. The LLC cannot resolve the address and the ARP functions hold the packet and generate an ARP request, passing it to the MAC layer after the 25 µs LLC processing delay, where it is queued behind 23 waiting TCP ACKs — packets 8016, 8020–8022, 8025–8027, 8029–8031, 8033, 8035, 8037, 8039, 8042–8044, 8047, 8048, 8050, 8055, 8058 and 8059, for segments 3997–4019.

Over the next 28 ms, `node_(0)` receives a further 23 packets (seq 4020–4042) from `node_(1)`, queues 23 more TCP ACKs (seq 4020–4042) and sends the 23 waiting TCP ACKs (seq 3997–4019) to `node_(1)`, before it can broadcast the ARP request to locate `node_(2)` at 10.02873176.

`node_(2)` is just 208 ns away and is the first to receive the 616 µs ARP request, which is passed to the ARP module. The ARP module processes the ARP request and returns a unicast ARP reply to its MAC layer after an LLC processing delay. The `node_(2)` MAC chooses a large number of slot times from the CW, while the `node_(1)` MAC chooses just seven slot times for its next queued packet and transmits packet 8086 (seq 4043) for `node_(0)` before `node_(2)` can send the ARP reply.

`node_(2)` sees the `node_(1)` transmission less than a microsecond later and pauses its backoff timer, calculating the number of whole idle-slots as ((current-time - (start-time + DIFS)) / SlotTime) truncating any fraction to the previous whole integer; and then decrementing the time remaining (rtime) by the product of idle-slots and SlotTime. After `node_(2)` successfully receives packet 8086 (`node_(1)` to `node_(0)`), it sets the NAV from the packet Duration field (set to the time to transmit a MAC ACK plus a SIFS).

The backoff timer remains paused while the MAC is not idle, with either physical carrier sense (transmitting or receiving) or virtual carrier sense (non-zero NAV).

Before the NAV expires and the backoff timer resumes, `node_(2)` begins receiving the MAC ACK from `node_(0)` to `node_(1)`. On completion of receiving the MAC ACK, `node_(2)` updates its NAV with the MAC ACK duration, always set to zero, and so resumes the backoff timer after a DIFS, setting the time remaining as (rtime + DIFS).

This process is repeated twice more as `node_(1)` sends another TCP data packet and `node_(0)` is also able to send a TCP ACK, both packets with their respective MAC ACK replies, before the `node_(2)` backoff timer finally expires and the ARP reply is sent to `node_(0)` 2,697.465 µs after the request.

The TCP SYN from `node_(0)` for `node_(2)` is addressed and queued behind the 24 waiting TCP ACKs (seq 4021–4044) for `node_(1)` on `node_(0)`.

After `node_(1)` begins to send its MAC ACK at 10.03112707, the last bit is sent 232 µs later at 10.03135907 and `node_(1)` waits a DIFS before an 11-slot contention backoff to send the next TCP data packet. The DIFS expires at 10.03140907 and the first 20 µs slot expires at 10.03142907 and `node_(1)` enters its second 20 µs slot. However, 159 ns later (the current propagation time from `node_(1)` to `node_(2)`[10]), at 10.03142923, `node_(2)` begins to transmit its ARP reply. This takes 159 ns to reach `node_(1)`, which pauses its backoff timer, calculating only one whole idle-slot as having expired. The ARP reply takes 129.1851852 µs to receive and `node_(2)` then waits the NAV from the packet duration (SIFS plus time to transmit a MAC ACK).

Before this expires, `node_(2)` sees the actual MAC ACK sent at 10.03156862 from `node_(0)`, 365 ns after it starts, and continues to pause its backoff timer. The MAC ACK takes a further 232 µs to receive and `node_(1)` then resets its NAV to zero, waits a DIFS and transmits the next packet after the remaining 10 slot times expire.

At 10.06436563, `node_(0)` has sent 24 more TCP ACKs up to seq 4044, but has also received 27 TCP data packets and queued TCP ACKs for seq 4045–4071, before it sends the TCP SYN and `node_(2)` prepares the SYN ACK. `node_(0)` sends TCP ACK 4045 to `node_(1)`, occupying the channel until `node_(1)` returns the MAC ACK to `node_(0)`.

Both `node_(2)` and `node_(0)` choose ten slots from the CW to transmit their queued packets. However, `node_(1)` is much closer to `node_(2)` than `node_(0)`. So `node_(2)` sees the end of the `node_(1)` transmission 206 ns before `node_(0)` does. Both nodes wait a DIFS and 10 slots from the time they received the last bit of the `node_(1)` transmission. At 10.06541842, `node_(2)` begins to transmit

[10]Propagation time determined from the 159 ns difference in the trace file between node_(1)'s slots (from the end of its ACK + DIFS) and node_(2)'s transmission.

the SYN ACK to node_(0), taking 208 ns to propagate to node_(0). However, only 206 ns later, at 10.06541862, node_(0) begins to transmit another TCP ACK to node_(1).

Just 2 ns after node_(0) begins to transmit the TCP ACK, it begins to receive the SYN ACK from node_(2), however *ns-2* fails to act on the error condition at node_(0) and the SYN ACK is processed. A MAC ACK is prepared, but the MAC is still waiting for a incoming MAC ACK response from node_(1) and the outgoing MAC ACK is delayed.

Meanwhile, node_(1) detects both packets and drops the incoming TCP ACK as a collision without acknowledgement. Eventually, node_(0) times out to retransmit the TCP ACK and now sends the delayed MAC ACK for its incorrectly received SYN ACK. However, node_(2) has already timed-out to retransmit the SYN ACK and drops this late acknowledgement as stale.

Soon after, node_(0) successfully retransmits the TCP ACK and, after many more packets transfer between node_(1) and node_(0), eventually node_(2) resends the SYN ACK, which node_(0) acknowledges again and this time drops as a duplicate.

Thus, besides the exacerbation of a collision during the initial TCP handshake, the second transfer has a very slow start as every packet from node_(0) gets queued behind 23–27 TCP ACKs for node_(1) that also contend with, on average, the same number of TCP data packets coming from node_(1) to node_(0). However, this roughly one-in-fifty window improves dramatically as the transfer gets under way and the TCP window expands on node_(0). With every TCP ACK received, node_(0) queues two more data packets, roughly doubling the transmit rate, as shown in Table 7.4, until the 20 packet *ns-2* default TCP window is filled, finally achieving roughly one quarter of the channel utilisation. Although node_(0) has equal probability of getting the channel when contending with node_(1), it is also sending TCP ACKs for node_(1), as well as the TCP data for node_(2).

Simultaneous Transmission Between Two Nodes

At time 10.229390899, a race condition presents in the simulation (although no race condition actually occurs in the deterministic execution of the simulator), as both node_(0) and node_(2) choose the same slot to send packets to each other and again the error in *ns-2* allows both to process the received packets,

Seq#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Pkts	52	47	3	56	2	5	3	53	2	7	4	5	1	6	1
Seq#	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Pkts	48	2	7	3/6	4	4	1	3	8	1	3	1	1	5	1

Table 7.4: Delay to Transmit TCP Data from node_(0) to node_(2)

even though each was transmitting at the same time, realistically destroying any received data.

Set-up of the Race Condition

At time 10.228848899±,[11] `node_(0)` begins to send the first bit of a 29-byte (232-bit) MAC ACK to `node_(1)`, in response to the last packet it received from `node_(1)`.

The propagation delay is calculated, with 52 bits of precision, for every node in range of `node_(0)` and a receive event is scheduled for every node at the appropriate time for that particular node. In this case, `node_(2)` is approximately 208 ns away and at 10.228849107 it will begin to receive the MAC ACK intended for `node_(1)`.

The MAC ACK is a control packet sent at 1 Mbps and takes 232 µs (232,000 ns) to transmit.

At 10.229080899, when the last bit is transmitted, `node_(0)` selects a DIFS plus 13 slots to backoff to transmit the next queued packet, TCP data seq 18 to `node_(2)`, and sets its backoff timer with 310 µs, to expire at 10.229390899.

At precisely the propagation time after the last bit is transmitted (approximately 208 ns), at 10.228849107, `node_(2)` begins to receive the first bit of the MAC ACK transmitted by `node_(0)`. At 10.229081107, when the last bit is received, `node_(2)` selects a DIFS plus 13 slots to backoff to transmit the next queued packet, TCP ACK seq 17 to `node_(0)`, and sets its backoff timer with 310 µs, to expire at 10.229391107.

At time 10.229390899, `node_(0)`'s backoff timer expires and it begins to send TCP data seq 18 to `node_(2)`. Again, propagation delay is calculated, with

[11]Although the *ns-2* scheduler uses doubles with 52 bits of mantissa, the trace file only displays time to nine decimal places (nano-seconds) and so any calculations shown here are only accurate to whole nanoseconds.

52 bits of precision, for every node in range of `node_(0)` and a receive event is scheduled for every node at the appropriate time for that particular node. Now `node_(2)` is scheduled to begin to receive TCP data seq 18 at exactly the same time it is already scheduled to transmit TCP ACK seq 17, at 10.229391107.

The Deterministic Simulation

The *ns-2* scheduler used here is a linked list of events. Where multiple events are scheduled at the same time, the ns-2 scheduler always executes events in the same order they were scheduled.

Thus, the transmit, scheduled after `node_(2)` saw `node_(0)`'s MAC ACK, will always execute before the receive, scheduled after `node_(0)` began sending the following packet.

At 10.229391107, `node_(2)` begins to transmit TCP ACK seq 17, setting `tx_state_ = MAC_SEND` and `tx_active_ = 1`, to `node_(0)`, which is actively transmitting TCP data seq 18 to `node_(2)`. At the same time, having already set its transmit state, it begins receiving TCP data seq 18, setting `rx_state_ = MAC_RECV`.

It takes `node_(2)` $133.\dot{9}2\dot{5}$ µs to transmit TCP ACK seq 17, after which it sets its `tx_active_ = 0`, but leaves `tx_state_ = MAC_SEND` while waiting for a 2 µs maximum propagation delay, a SIFS, a return MAC ACK and a second 2 µs maximum propagation delay; and is still receiving the TCP data seq 18.

At 10.229525241, `node_(0)` finishes receiving TCP ACK seq 17 and processes it, but cannot send a MAC ACK while it is still transmitting TCP data seq 18.

At 10.229600677, `node_(0)` finishes transmitting TCP data seq 18, and begins to wait for a 2 µs maximum propagation delay, a SIFS, a return MAC ACK and a second 2 µs maximum propagation delay.

At 10.229600885, `node_(0)` finishes receiving TCP data seq 18 and processes it, but cannot send a MAC ACK while it is still `tx_state_ = MAC_SEND`, waiting for the MAC ACK from `node_(2)`.

At 10.229771033 (10.229391107 + $133.\dot{9}2\dot{5}$ µs + 2 µs + 10 µs + 232 µs + 2 µs), `node_(2)` times-out sets to retransmit TCP ACK seq 17, waits a SIFS and begins to send its late MAC ACK to `node_(0)` at 10.229781033, once again taking 208 ns to propagate to `node_(0)` and 232 µs to receive the 232 bits at 1 Mbps.

Only 75.644 µs later, at 10.229846677, 10.229600677 + 246 µs, `node_(0)` times-out to retransmit TCP data seq 18. The IEEE 802.11 protocols do not check carrier-sense when sending the MAC ACK, so `node_(0)` waits a SIFS and sends its late MAC ACK to `node_(2)` at 10.229856677, destroying the incoming (also late) MAC ACK from `node_(2)`. At 10.230088885, 232.208 µs later, `node_(2)` finishes receiving the late MAC ACK and drops it as stale.

Next `node_(2)` retransmits the unacknowledged TCP ACK seq 17, which `node_(0)` acknowledges and drops as a duplicate; and `node_(0)` retransmits the unacknowledged TCP data seq 18, which `node_(2)` acknowledges and drops as a duplicate.

As the number of contending nodes increases, the potential and likelihood of these and similar collisions also increases, wasting bandwidth and increasing individual contention windows, reducing the overall throughput. However, as the number of contending nodes increases, the probability of *all* nodes choosing a long contention backoff, at any given time, from their individual contention windows, decreases; and the probability of *at least one* node choosing a very short contention backoff increases; resulting in shorter contention backoffs, increasing the overall throughput.

Thus, increasing the number of nodes increases the overall channel throughput, even though the throughput of individual nodes deceases — up to the point at which the time lost to collisions exceeds the time gained by shorter contention backoffs.

Recurring Patterns in the Remainder of the Simulation

This same pattern of slow-start, slowly rising to a steady shared channel utilisation is repeated for every new transfer initiating from the already-fully-queued AP. The first data packet for `node_(4)` is not successfully transferred until over 3 seconds after the FTP session initiates.

Conversely, each new transfer from another node to the AP does not suffer the same severity of this slow start, as there is no existing queue on the sending node — although enlargement of the TCP window does require waiting for queued TCP ACKs from the AP.

Initially, the first node is able to effect almost three quarters of the total throughput, effectively having to contend only with the AP for the channel and the AP has to send all the TCP ACKs from the first transfer, as well as any TCP data for the second transfer. However, as the simulation progresses, and the AP has multiple concurrent transfers, the node transferring data to the AP is more and more likely to have to contend with one of the other nodes returning TCP ACKs to the AP and this advantage reduces to two thirds of the total transfer rate — still considerably more than the sum of all the transfers outbound from the AP, which equally share the remaining one third of the throughput.

Table 7.5 provides a graphic of the proportion of the total throughput achieved by each of the transfers at various stages throughout the simulation.

Time	1–0	0–2	3–0	0–4	5–0	0–6	7–0	0–8	9–0	0–10
10.5–11.5	74%	26%								
15.5–23		28%	72%							
23.5–24		19%	71%	10%						
25.5–30				29%	71%					
30.5–34				16%	69%	14%				
34.5–35				50%		50%				
35.5–40				19%		12%	69%			
40.5–45.5				9%		13%	68%	10%		
45.5–50				12%		12%		11%	66%	
50.5–55				9%		8%		8%	66%	10%
55.5–56.5				18%		39%		22%		21%
57–64.5						32%		30%		37%
65–69.5								43%		57%

Table 7.5: Proportion of the Channel Throughput for each Transfer

In this case, the ten 2.6 MB FTP sessions take 65 seconds complete, transferring 2,905,248 bytes each — a 60-byte TCP SYN plus 5,079 * 572-byte TCP data packets (2,905,820 bytes for `node_(4)`, with an extra 572-byte packet).

The individual performance of each connection is summarised in the following Table 7.6. For every connection, the average steady throughput of the previous non-overlapping TCP simulation is compared against the average throughput in this overlapping TCP simulation for the same periods, then the best average steady throughput for the connection in this overlapping simulation is given, discounting as many as possible overlapping or trailing time periods. The average of all ten of these best average steady throughputs was 2.569 Mbps (0.3211 MBps), down from the 3.611 Mbps (0.4514 MBps) of the non-overlapping simulation, however the overall average throughput, was $\frac{29{,}053{,}052*8}{65.0} = 3,575,760.2$ bps or 3.576 Mbps or 0.4470 MBps, the same as the non-overlapping simulation.

This simulation was also executed recording the total throughput of all nodes in a single trace. The results are shown in Figure 7.11.

Link	Previous Time	Non-overlap	Overlapping	Steady-state	Throughput
1–0	5.0–9.5	3,683,278.$\dot{2}$	3,683,278.$\dot{2}$	5.0–10.0	3,678,284.8
0–2	10.0–15.0	3,468,704.0	2,541,606.4	12.0–15.0	3,460,981.$\dot{3}$
3–0	15.5–20.0	3,678,087.$\dot{1}$	2,588,999.$\dot{1}$	15.5–23.0	2,594,897.1
0–4	20.5–24.5	3,677,960.0	168,288.0	24.5–25.0	3,102,528.0
5–0	25.0–29.5	3,668,024.$\dot{8}$	2,450,808.$\dot{8}$	25.5–30.0	2,591,032.$\dot{8}$
0–6	30.0–35.0	3,505,312.0	678,259.2	34.5–35.0	1,812,096.0
7–0	35.5–40.0	3,584,533.$\dot{3}$	2,493,411.$\dot{5}$	36.0–40.0	2,535,104.0
0–8	40.5–44.5	3,652,792.0	367,224.0	65.0–69.5	1,534,485.$\dot{3}$
9–0	45.0–50.0	3,514,464.0	2,231,353.6	45.5–50.0	2,355,114.$\dot{6}$
0–10	50.5–54.5	3,679,104.0	351,208.0	65.0–69.5	2,025,642.$\dot{6}$
All	Avg above	3,611,225.9$\dot{5}$	1,755,443.7	Avg above	2,569,016.7
All	All (5.0–55.0)	3,575,782.4		All (5.0–70.0)	3,575,760.2

Table 7.6: Comparing Non-overlapping and Overlapping TCP Throughput

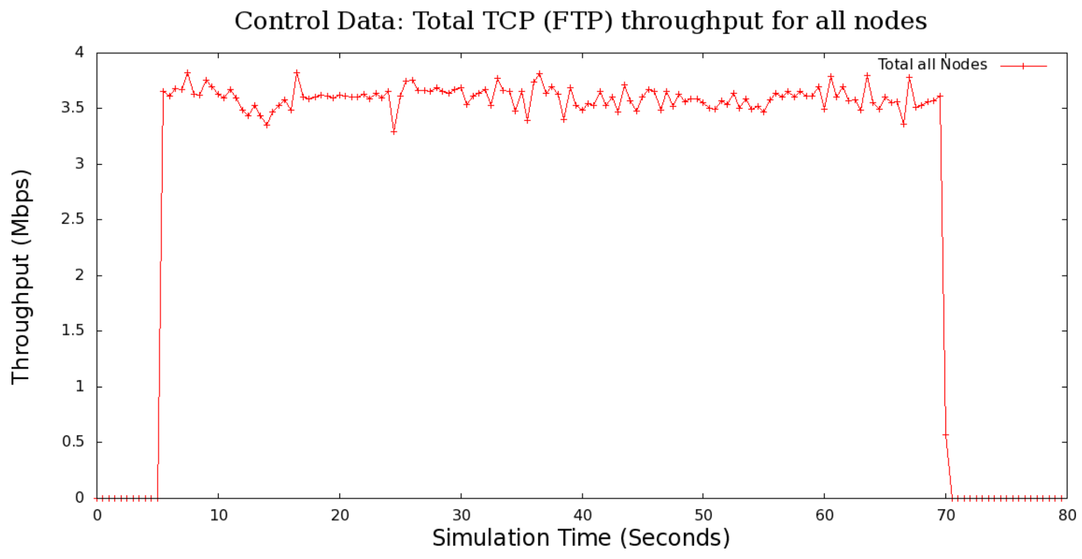


Figure 7.11: Control Data: Total TCP throughput (Overlapping FTP) all nodes

Figure 7.11 shows the total throughput at any given time for all sending STAs in a single trace. This was performed using the same scenario, as the previous simulation and gives the overall average throughput, for the 65 seconds, as 3,575,760.2 bps or 3.576 Mbps, identical to the previously calculated results and again the same as the non-overlapping simulation.

7.2 Testing the Proposals

With the control data now established and thoroughly analysed in its own right, the author then modified the *ns-2* simulator to implement the proposals. The various approaches to testing new protocols are now discussed and the chosen approach is then detailed.

7.2.1 Adding New Protocols to ns-2

The accepted method for modifying or developing new agents and protocols within the *ns-2* simulator, was to modify the original source code, or to copy components of the source code into new files and then modifying the new files as additions to the original source. This was then inserted into the *ns-2* source tree and the entire tree recompiled to produce a new simulator executable containing the altered or additional functionality.

In this manner, developing new protocols could be considerably time-consuming and resource-hungry, particularly where a series of tests and modifications were required in the normal course of the development cycle. As such, development of new protocols for the simulator could be a somewhat slow and tedious process, interspersed with frequent breaks while the source tree was compiled.

Moreover, where such modifications prove popular and require distribution, these would then need to be distributed either as patches to a particular revision of the source code, or as an entire special revision of the source distribution itself. Persons wishing to use these modifications, would at a minimum need to download the appropriate patch for their *ns-2* revision, patch the source and then recompile it. At worst, it may be necessary to download an entire special *ns-2* distribution to achieve their goals.

7.2.2 Dynamic Modules in ns-2

Federico Maguolo and Nicola Baldo from the Special Interest Group on NETworking (SIGNET) in the Department of Information Engineering at the University of Padova designed and developed patches for *ns-2* to utilise the ability to load dynamic libraries in Tcl as a way of developing extensions to *ns-2*.

Maguolo's patches to allow dynamically loadable libraries considerably speeds development time for *ns-2* experiments. New agents and new protocols can be developed on the fly without the need to modify the simulator source code, nor to recompile the entire simulator. Using these patches, the dynamic libraries are developed as independent modules and are loaded as part of the simulation at the time, rather than having to be a permanent part of the simulator itself.

This also provides a vector for the distribution of the popular extensions, as such dynamically loadable libraries can be sourced and loaded at simulation time by those who desire to use them. Furthermore, such modifications will not affect the compatibility of the core simulator program.

Maguolo's patches modified the core *ns-2* simulator to allow the dynamic extension of the declared types within the various headers, thus allowing any additional modules to provide definitions of new types of agents, protocols and so forth. Maguolo's patches were developed for *ns-2.29* and *ns-2.31* and were incorporated into the codebase of *ns-2.33*.

The dynamically loadable libraries were tested with the SIGNET sample dynamic libraries, in this case, the dynamic library tcp-veno package available from http://www.dei.unipd.it/ baldo/tcp-veno-1.0.2.tar.gz The following steps were performed:

```
cd /usr/src
wget http://www.dei.unipd.it/~baldo/eurane-umts-1.10.2.tar.gz
wget http://www.dei.unipd.it/~baldo/tcp-veno-1.0.2.tar.gz
tar xzvf tcp-veno-1.0.2.tar.gz
cd tcp-veno-1.0.2
./configure --with-ns-allinone=/usr/src
make
cd samples
ns test-veno.tcl
```

The test, having completed successfully, confirmed that dynamically loadable libraries were operational in the *ns-2* simulator tool.

7.2.3 Developing the New Dynamic Modules

The development of new dynamic modules to implement the proposed WLS and PWLS protocols began with the setting up of the development environment as follows.

```
cd ~/ns
mkdir drproto
cd drproto
mkdir src
mkdir m4
```

Next, a complete set of autotools macros for dynamically loadable libraries for *ns-2* was downloaded from under Nicola Baldo's University of Padova Web page [181], at http://www.dei.unipd.it/˜baldo/nsallinone.m4 into the m4 subdirectory and the autogen script was created in the parent directory, as follows.

```
cd m4
wget http://www.dei.unipd.it/\~baldo/nsallinone.m4
cd ..
vi autogen.sh
```

Append the following two lines [181]:

```
#!/bin/sh
aclocal -I m4 --force && libtoolize --force && automake --foreign
--add-missing && autoconf
```

Make it executable and create configure.ac

```
chmod a+x autogen.sh
vi configure.ac
```

Create configure.ac with the following [181]:

```
AC_INIT(foo,1.0)
AM_INIT_AUTOMAKE
AC_PROG_CXX
AC_PROG_MAKE_SET
AC_DISABLE_STATIC
AC_LIBTOOL_WIN32_DLL
AC_PROG_LIBTOOL
```

```
AC_PATH_NS_ALLINONE
AC_DEFINE(CPP_NAMESPACE,std)
AC_CONFIG_FILES([
    Makefile
    src/Makefile
    m4/Makefile
    ])
AC_OUTPUT
```

Create Makefile.am in this parent directory with the following [181]:

```
SUBDIRS = src m4
EXTRA_DIST = autogen.sh
ACLOCAL_AMFLAGS = -I m4
DISTCHECK_CONFIGURE_FLAGS = @NS_ALLINONE_DISTCHECK_CONFIGURE_FLAGS@
```

Create Makefile.am in the m4 directory with [181]:

```
EXTRA_DIST = nsallinone.m4
```

Create Makefile.am in the src directory with [181]:

```
lib_LTLIBRARIES = libfoo.la

libfoo_la_SOURCES = <list of sources>
libfoo_la_CPPFLAGS = @NS_CPPFLAGS@
libfoo_la_LDFLAGS =  @NS_LDFLAGS@
libfoo_la_LIBADD =   @NS_LIBADD@



nodist_libfoo_la_SOURCES = embeddedtcl.cc
BUILT_SOURCES = embeddedtcl.cc
CLEANFILES = embeddedtcl.cc

TCL_FILES =  foo-init.tcl

embeddedtcl.cc: Makefile $(TCL_FILES)
    cat $(TCL_FILES) | @TCL2CPP@ FooTclCode > embeddedtcl.cc

EXTRA_DIST = $(TCL_FILES)
```

In the process of developing these files, it was soon discovered, that in order to develop a dynamically loadable library to perform the `Mac/802_11` functions using WLS or PWLS, that the entire IEEE 802.11 MAC layer and channel protocols would have to be duplicated and given a unique name to allow other inclusion as a dynamic library without interfering with the existing functionality.

At this point it was realised that the advantages of dynamically loadable libraries are only valid when a substantially new protocol or agent is being simulated.

In this case, as we are wishing to simulate modifications to an existing extensive set of protocols and agents, it is more efficient to directly modify copies of the existing code and recompile the *ns-2* simulator tool. The development of dynamically loadable libraries was abandoned in favour of duplicating existing elements within the `Mac/802_11` protocols and making our modifications directly to the source. Once modified, this alternate code was then recompiled with the remainder of the source to produce a new version of the *ns-2* simulator tool.

7.2.4 Class Structures for ns-2 Mobile Nodes

Mobile nodes contain address and port demultiplexers to source or sink agents, a routing agent, an LLC component to handle sequence numbers and link addressing with an ARP component to resolve for MAC addresses, feeding an interface queue into the MAC component and the network interface into the channel.

The class `MobileNode` simulates the wireless STAs themselves. The class includes position, speed and energy states; and can calculate distance and propagation delays to other mobile nodes. The implementation of the class was not modified during this work. The code is located in the files *~ns2/common/mobilenode.h* and *~ns2/common/mobilenode.cc*.

The class `LL` simulates the LLC upper sub-layer of the link layer and while the code itself remained unchanged in files *~ns2/mac/ll.h* and *~ns2/mac/ll.cc*, the operation of the LLC was modified through the *OTcl—C++* variable bindings. It includes the variables:

- `bandwidth_` : which is not used in the `LL` class, as previously discussed; and
- `delay_` : the link (in this case, LLC) delay, which is inherited from the parent `LinkDelay` subclass of the `Connector` class and set by default to 25 µs.

The class `LL` manages the `arptable_`, schedules `UP` packets to go to the `uptarget_` after the `delay_` and determines the MAC address (but does not encapsulate it) and schedules `DOWN` packets to go to the `downtarget_` after the `delay_`.

The class `PriQueue` is used to simulate the link layer queue which gives priority to routing packets and remains unchanged in files *~ns2/queue/priqueue.h* and *~ns2/queue/priqueue.cc*.

The class `Mac802_11` simulates a (mostly IEEE 802.11b) wireless MAC and has source code in files *~ns2/mac/mac-802_11.h* and *~ns2/mac/mac-802_11.cc*, modified as described below.

The class `WirelessPhy` simulates the interface to wireless physical layer and remains unchanged in files *~ns2/mac/wireless-phy.h* and *~ns2/mac/wireless-phy.cc*.

The class `WirelessChannel` simulates the actual transmission of the packet at the physical layer and remains unchanged in files *~ns2/mac/channel.h* and *~ns2/mac/channel.cc*.

7.3 Testing Partial WLS

Although "Partial WLS", with the destination address unencrypted, was only an alternate to the primary objective of WLS itself, with the entire MPDU encrypted, Partial WLS is the easiest to implement and involved the least uncertainty; and so was the first to be simulated.

7.3.1 Configuring Partial WLS

As Partial WLS does not encrypt the destination address, no modifications were required to the MAC address filtering or handling routines. The principal consideration with Partial WLS was the delay in encrypting an outgoing packet before passing it to the lower layers; and the delay in decrypting a packet addressed for the current node before passing it to the higher layers.

Initially this was performed by changing the LLC processing delay to simulate longer handling times going down to or up from the MAC sub-layer.

The initial value for an encryption cost consisted of a relatively slow base cost of 0.9 µs + 0.009 µs per byte. In our simulations with a maximum of 2304-byte packets plus the extra 20 bytes from the DSDV routing not being removed would therefore transfer maximum 2324-byte packets. These induce a maximum encryption/decryption cost of 21.816 µs.

To represent this additional delay in encrypting an outgoing packet before passing it to the lower layers and the delay in decrypting an incoming packet before passing it to the higher layers, this encryption/decryption cost was added to the default LLC processing delay of 25 µs to give a total delay of 46.816 µs, as shown below.

The simulation of *drtestg.tcl*, producing the control data in Figure 7.1, was reconfigured with this additional delay in *drtestr.tcl*, as follows:

```
Agent/UDP set packetSize_ 2304
Agent/TCP set packetSize_ 2304

# Standard Link Layer delay 25us PLUS
# PARTIAL WLS (21.816us to decrypt frame but destination is understood)
LL set delay_ 46.816us

Mac/802_11 set dataRate_ 54Mb
Mac/802_11 set RTSThreshold_ 3000
Mac/802_11 set PreambleLength_ 72
```

7.3.2 Unexpected Results

The results are shown in Figure 7.12.

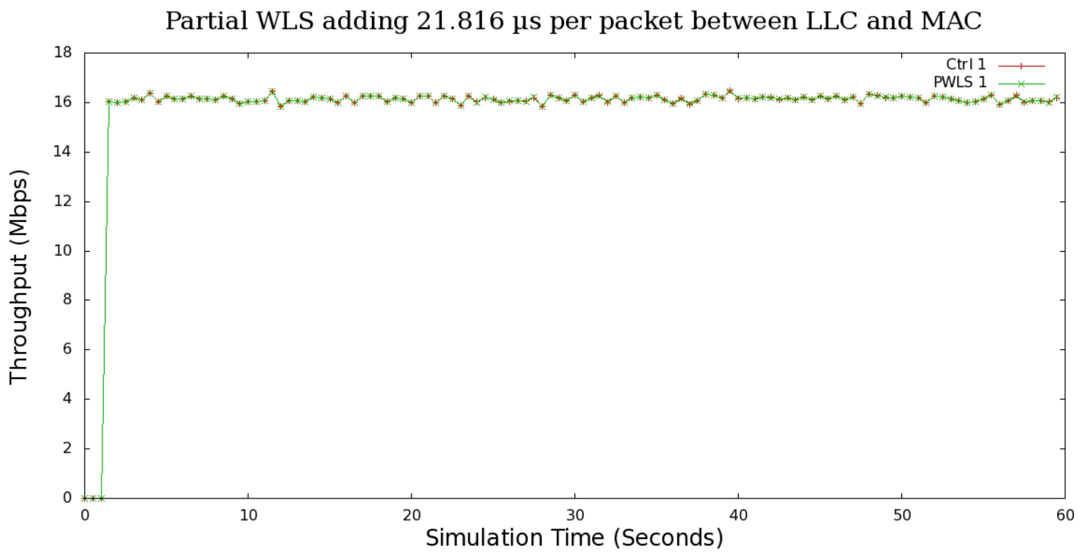


Figure 7.12: PWLS simulated by adding extra delay between the LLC and MAC

Figure 7.12 shows the CBR generator of `node_(1)` plotted over the original `node_(1)` from Figure 7.1, starting at time 1.0 and initially achieving an average throughput of 16.01 Mbps in the first second of operation, identically to the previous scenarios and the control data in Figure 7.1, consolidating to a 16.15 Mbps average throughput over the simulation.

These results are virtually identical to the control data showing that Partial WLS has no effect on throughput in this two-node scenario.

The author was not expecting these results, gathered before the detailed analysis of the empirical data, having assumed that the cost of encryption would have to have some effect on the throughput. It was considered that these results may be distorted, with the effects masked, because the encryption delay had been confined entirely to the LLC sub-layer; and thus more accurate results may issue if the simulation were to more precisely represent the physical processes.

7.3.3 Refining the Tools

The code was reviewed tracing the path of the "packet" through the various components of the nodes. The LLC delay was reset to the default 25 µs and the cost of encryption was moved into the MAC sub-layer by modifying the `Mac802_11` class.

To ensure a reasonable simulation of a practical real-world encryption engine, the author used the specifications of Sivakumar and Velmurugan's "High Speed VLSI Design CCMP AES Cipher for WLAN (IEEE 802.11i)" [182], as a leading-edge hardware cryptographic engine for WLAN use, providing throughputs of 2.257 Gbps for encryption and 1.722 Gbps for decryption.

Adding Encryption Delays in the ns-2 MAC Sub-layer

A new subclass `CryptoTimer(Mac802_11 *m)` of the `MacTimer(m)` class was added to the `MacTimer` class header file, *~ns2/mac/mac-timers.h*, as shown below.

```
class CryptoTimer : public MacTimer {
public:
    CryptoTimer(Mac802_11 *m) : MacTimer(m) {}
    void    handle(Event *e);
};
```

The matching handler stub code was added to the `MacTimer` class *C++* source file, *~ns2/mac/mac-timers.cc*, as shown here.

```
/* ================================================================
   Crypto Timer to encrypt/decrypt entire frames
   ================================================================ */
void
CryptoTimer::handle(Event *)
{
    busy_ = 0;
    paused_ = 0;
    stime = 0.0;
    rtime = 0.0;

    mac->cryptoHandler();
}
```

The `Mac802_11` class header file, *~ns2/mac/mac-802_11.h*, was then modified to add the new timer `friend class CryptoTimer`, handled by the `void cryptoHandler(void)`, passing the outgoing copy of the packet in the structure pointed to by `Packet *delayedPacket_`. These declarations were then used in the modification of the `Mac802_11` class *C++* source file, *~ns2/mac/mac-802_11.cc*, to insert the appropriate cost of encryption.

Encryption Pipelining in the MAC Sub-layer

It was assumed a pipelining process would be used, so that the start of the ciphertext can be transmitted while the remainder of the stream is still being encrypted.

In the unlikely event that the encryption rate is slower than the data rate, then sufficient units of the frame must be encrypted before the transmission begins, so that the encryption of the last unit has completed before it is to be transmitted. This adds a delay before transmission can begin. However, transmission and propagation themselves are unaffected and proceed with the normal timings. The worst case here is an encryption rate so slow that the entire frame has to be encrypted before transmission begins.

It was assumed that normally the encryption rate will be very much faster than the data rate. This is reasonable given the current relative states of both the encryption and IEEE 802.11 transmission technologies. While this may come into question should advances in multi-gigabit over-the-air data rates outstrip advances in encryption rates, there is sufficient margin to assume any such advances will even out in the long term. As such, the only delay in this case is the encryption of the first ciphertext unit, which must occur before transmission can begin. Assuming a worst case of a 128-bit block cipher in Cipher-Block-Chaining (CBC) mode, this would require the first 128-bit encryption to be completed before transmission begins.

The ns-2 Mac802_11 Class

The `Mac802_11` class, from the *C++* source file, *~ns2/mac/mac-802_11.cc*, provides the various methods for the MAC sub-layer, these include:

- `Mac802_11::recv(Packet *p, Handler *h)` : receives a packet from the LLC queue or starts to receive an incoming packet from the wireless channel — passes downward-going packets to `send(p, h)`, otherwise checks for collisions and starts the `mhRecv_` timer to wait `txtime(p)`;
- `Mac802_11::recvHandler()` : the `mhRecv_` timer handler stub (finished receiving a frame), calls `recv_timer()`;
- `Mac802_11::recv_timer()` : the `mhRecv_` timer handler, which ignores received packets if still transmitting when the receive finishes, drops collisions and error packets setting the NAV with an EIFS, updates the NAV for non-local-destination packets, discards non-local-destination directed packets, passes MAC ACKs to `recvACK(pktRx_)` or data packets to `recvDATA(pktRx_)` and calls `rx_resume()`;
- `Mac802_11::recvACK(Packet *p)` : if not `MAC_SEND` (waiting for an ACK) drops the packet as `DROP_MAC_INVALID_STATE`, otherwise stops the `mhSend_` timer (to stop a retransmit), resets the retry count and the `cw_`, removes the previous packet from the queue and, if another is waiting, starts the `mhBackoff_` timer with the reset `cw_`, and calls `tx_resume()`;
- `Mac802_11::recvDATA(Packet *p)` : strips of the length of the PLCP and MAC headers, sends a MAC-layer ACK and calls `tx_resume()` for directed

packets, checks the sequence number and drops duplicates, copies broadcasts to the `uptarget_->recv(p->copy(), (Handler*) 0)`, passes broadcasts and packets for other nodes back to the channel interface and sends local-destination packets to the `uptarget_->recv(p, (Handler*) 0)`;

- `Mac802_11::send(Packet *p, Handler *h)` : calls `sendDATA(Packet *p)` to add the headers and calculate the `txtime()`, assigns a sequence number and, if not already in backoff, starts `mhBackoff_` with an initial DIFS wait if the medium is idle or no initial wait if not idle;
- `Mac802_11::sendDATA(Packet *p)` : adds the length of the PLCP and MAC headers, adds the applicable header details and calculates the `txtime()` and `dh_duration`;
- `Mac802_11::deferHandler()` : the `mhDefer_` timer handler stub, which calls `check_pktCTRL()` for waiting MAC ACKs or calls `check_pktTx()` for waiting data packets;
- `Mac802_11::backoffHandler()` : the `mhBackoff_` timer handler stub, which calls `check_pktTx()` for waiting data packets;
- `Mac802_11::check_pktCTRL()` : does not check the medium idle state for ACKs, sets the state `MAC_ACK`, sets the transmission `timeout` and calls `transmit(pktTx_, timeout)`;
- `Mac802_11::check_pktTx()` : if the medium is not idle, increments the `cw_` and starts the `mhBackoff_` timer again, otherwise sets the state `MAC_SEND`, sets the transmission `timeout` and calls `transmit(pktTx_, timeout)`;
- `Mac802_11::transmit(Packet *p, double timeout)` : sets an error state on any incoming packets [this does not work], copies (so as to retransmit later) the packet to the `downtarget_->recv(p->copy(), this)`, starts the `mhIF_` timer with the `txtime(p)` for the busy channel interface and starts the `mhSend_` timer with the `timeout`;
- `Mac802_11::txHandler()` : the `mhIF_` timer handler, which resets the interface transmitter active state after the frame is sent;
- `Mac802_11::sendHandler()` : the `mhSend_` timer handler stub (send `timeout` expired), which calls `send_timer()`;

- `Mac802_11::send_timer()` : if sending an ACK, frees the local resources, if sending a data packet, calls `RetransmitDATA()`, and calls `tx_resume()`;
- `Mac802_11::RetransmitDATA()` : if sending a broadcast (doesn't get ACKed), frees the local resources, resets the `cw_` and starts the `mhBackoff_` timer, otherwise increments the error counts, if the retry count is exceeded, drops the packet as `DROP_MAC_RETRY_COUNT_EXCEEDED` and resets the retry count and the `cw_`, otherwise increments the `cw_` and starts the `mhBackoff_` timer again;
- `Mac802_11::rx_resume()` : sets interface receive state to `MAC_IDLE` and pauses/resumes any `mhBackoff_` timer according to overall medium state;
- `Mac802_11::tx_resume()` : if waiting to send an ACK or a unicast packet without RTS not already in backoff, starts the `mhDefer_` timer for a SIFS, otherwise if not already in backoff, starts the `mhBackoff_` timer with an initial DIFS, sets interface transmit state to `MAC_IDLE` and pauses/resumes any `mhBackoff_` timer according to overall medium state;
- `Mac802_11::navHandler()` : the NAV handler, which resumes `mhBackoff_` after a DIFS;

Mac802_11 Class Modifications

The `Mac802_11` class was first modified at the point where it passes the frame to the channel interface. The original code is shown below.

```
inline void
Mac802_11::transmit(Packet *p, double timeout)
{
...
    downtarget_->recv(p->copy(), this);
    mhSend_.start(timeout);
    mhIF_.start(txtime(p));
}
```

This was modified to schedule the copy of the packet to be handled by a CryptoTimer() after the 0.00000005671245 seconds it takes to encrypt the first 128-bit block, as shown following.

```
inline void
Mac802_11::transmit(Packet *p, double timeout)
{
...
    /*
     * downtarget_->recv(p->copy(), this);
     * Crypto cost on sending a packet is just the encryption
     * of the maximum 2324-octet frame
     * at 2324 * 8 / 2257000000 = 0.000008237483 seconds
     * which is far less than the MAC data rate 54Mb = 0.000344296
     * So, the only delay is encrypting the first 128 bit block
     * 128 / 2257000000 = 0.00000005671245 seconds
     */
    delayedPacket_ = p->copy();
    Handler *postdecrypt_ = new CryptoTimer(this);
    // Scheduler::instance().schedule(postdecrypt_, delayedPacket_, 0.0);
    Scheduler::instance().schedule(postdecrypt_, delayedPacket_, 0.00000005671245);
    mhSend_.start(timeout);
    mhIF_.start(txtime(p));
}
...
void
Mac802_11::cryptoHandler()
{
    /*
     * Crypto cost on sending a packet
     */
    downtarget_->recv(delayedPacket_, this);
}
```

The simulator was recompiled and a new simulation of *drtestr.tcl* (from *drtestg.tcl*), was reconfigured with the original default LLC delay of 25 µs in *drtests.tcl*, as follows:

```
LL set delay_ 25us
...
    set f_($i) [open drtest$t-encr$i.tr w]
...
    exec xgraph drtestg-node1.tr drtest$t-encr1.tr -geometry 800x400 &
```

7.3.4 Results Confirmed

This simulation produced the same results as the previous method of *drtestr.tcl*, shown in Figure 7.12, with throughput following the original control data in Figure 7.1.

These results confirmed that, using reasonable specifications, Partial WLS has no effect on throughput in this two-node scenario.

In an attempt to quantify the magnitude of this immunity to encryption delays, the author completed a series of further simulations, steadily increasing the encryption delay and recompiling the simulator until a deviation was achieved. This occurred at a factor of 30 times the original encryption delay, as shown in the code below and in Figure 7.13.

```
inline void
Mac802_11::transmit(Packet *p, double timeout)
{
...
    /*
     * downtarget_->recv(p->copy(), this);
     * Crypto cost on sending a packet is just the encryption
     * of the maximum 2324-octet frame
     * at 2324 * 8 / 2257000000 = 0.000008237483 seconds
     * which is far less than the MAC data rate 54Mb = 0.000344296
     * So, the only delay is encrypting the first 128 bit block
     * 128 / 2257000000 = 0.00000005671245 seconds
     */
    double crptdelay = 30.0 * 0.00000005671245;
    delayedPacket_ = p->copy();
    Handler *postdecrypt_ = new CryptoTimer(this);
    // Scheduler::instance().schedule(postdecrypt_, delayedPacket_, 0.0);
    // Scheduler::instance().schedule(postdecrypt_, delayedPacket_, 0.00000005671245);
    Scheduler::instance().schedule(postdecrypt_, delayedPacket_, crptdelay);
    mhSend_.start(timeout);
    mhIF_.start(txtime(p));
}
```

This gives a transmission delay of 1.7 µs and represents an encryption rate of only 75.2$\dot{3}$ Mbps. The results are shown in Figure 7.13. While the encryption rate here has not yet dropped below the data rate, the additional delay was now causing the send timeout, based on a maximum 2 µs propagation delay, to be exceeded for most packets.

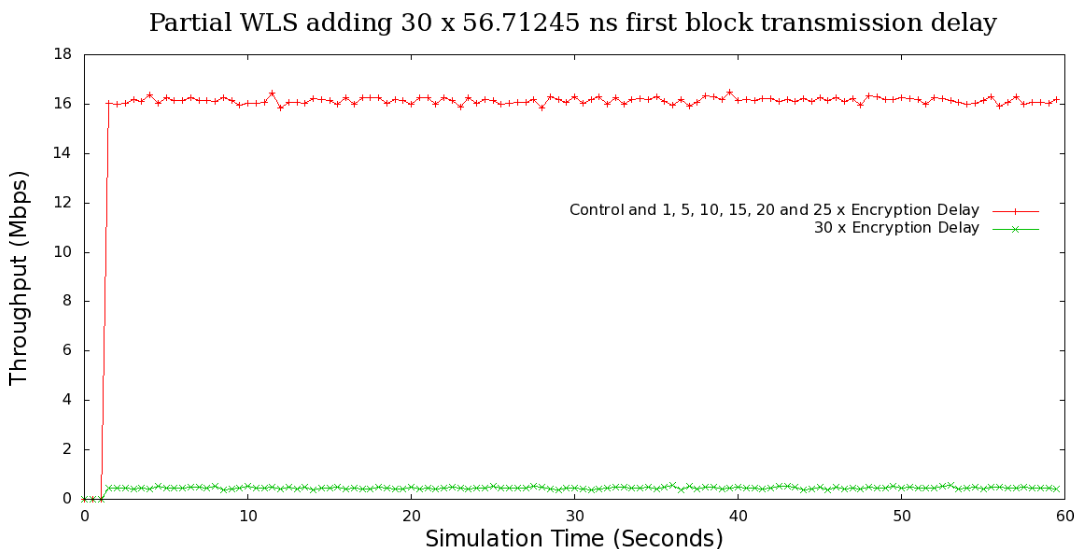


Figure 7.13: PWLS simulated by adding block encryption transmission delay

To extend this, the effects of decrypting a received packet in the MAC were added to the modified simulator code. A new subclass `DstDecTimer()` of the `MacTimer()` class was added to the `MacTimer` class header file, *~ns2/mac/mac-timers.h*.

```
class DstDecTimer : public MacTimer {
public:
    DstDecTimer(Mac802_11 *m) : MacTimer(m) {}
    void    handle(Event *e);
};
```

The matching handler stub code was added to the `MacTimer` class *C++* source file, *~ns2/mac/mac-timers.cc*, as shown below.

```
/* ============================================================
   DstDec Timer to decrypt dest addr with every available key
   ============================================================ */
void
DstDecTimer::handle(Event *)
```

```
{
    busy_ = 0;
    paused_ = 0;
    stime = 0.0;
    rtime = 0.0;

    mac->dstDecHandler();
}
```

The `Mac802_11` class header file, *~ns2/mac/mac-802_11.h*, was again modified to add the new timer `friend class DstDecTimer`, handled by the `void dstDecHandler(void)`. These declarations were then used in the modification of the `Mac802_11` class *C++* source file, *~ns2/mac/mac-802_11.cc*, to insert the appropriate cost of encryption at the point where it passes the frame to the LLC interface. The original code is shown below.

```
Mac802_11::recvDATA(Packet *p)
{
...
    /*
     *  Pass the packet up to the link-layer.
     *  XXX - we could schedule an event to account
     *  for this processing delay.
     */
...
    uptarget_->recv(p, (Handler*) 0);
}
```

This was modified to schedule the copy of the packet to be handled by a DstDecTimer() after the 0.00000007433217 seconds it takes to decrypt the first 128-bit block, shown here with the same multiple of 30 applied to produce an effect in the results, as shown in Figure 7.14.

```
Mac802_11::recvDATA(Packet *p)
{
...
    /*
     * uptarget_->recv(p, (Handler*) 0);
     * Decrypting the maximum 2324-octet frame for Partial WLS
     * at 2324 * 8 / 1722000000 = 0.000010796748 seconds
     * which is far less than the MAC data rate 54Mb = 0.000344296
     * So, the only delay is encrypting the first 128 bit block
     * 128 / 1722000000 = 0.00000007433217 seconds
     */
    double addrdelay = 30.0 * 0.00000007433217;
    delayedPacket_ = p;
    Handler *postdecrypt_ = new DstDecTimer(this);
    // Scheduler::instance().schedule(postdecrypt_, delayedPacket_, 0.00000007433217
    Scheduler::instance().schedule(postdecrypt_, delayedPacket_, addrdelay);
}
...
void
Mac802_11::dstDecHandler()
{
    /*
     * Decrypting the frame for Partial WLS
     */
    uptarget_->recv(delayedPacket_, (Handler*) 0);
}
```

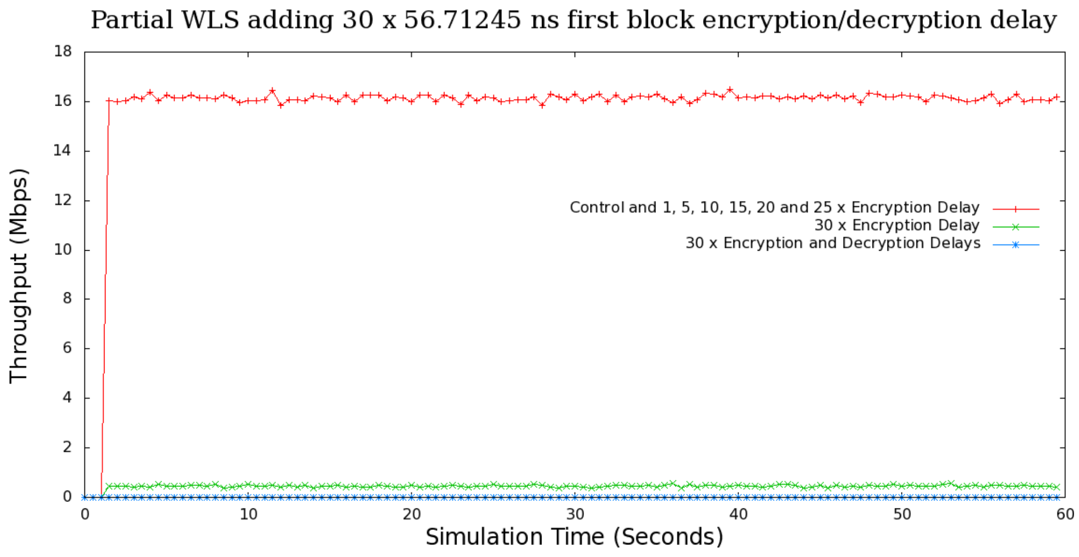


Figure 7.14: PWLS adding both encryption and decryption delays

7.4 Testing WLS

The tools were further modified to simulate 'full' WLS, with the destination address encrypted. This presents no new difficulties for the sending STA, however the receiving STAs must begin to decrypt the headers of every packet they see, in order to determine the destination address and therefore to know whether or not the packet was intended for them.

7.4.1 Configuring WLS

As PWLS and WLS employ effectively the same process for the sending STA in this respect, no modifications were required to the MAC `transmit()` routine. The principal consideration with full WLS was the effects of delays in decrypting the destination address in the `recv_timer()` routine that filters non-local addresses and calls the various helper routines, based on the packet type, and subsequently, the `recvDATA()` routine that processes incoming data packets.

Although the header processing functionality is split across two routines in the ns-2 `Mac802_11` class, the `recvDATA()` is called only by the `recv_timer()` routine and so is functionally all one routine. This simplifies the approach, as the decryption cost can be implemented in the one place, at the point where `recvDATA()` prepares to pass the packet to the `uptarget_`.

The decryption cost varies between ordinary STA and APs. An ordinary STA in a BSS will typically have only one pairwise key between itself and the AP. As such, if it can successfully decrypt a packet then the packet was indeed from the AP and destined for that STA. However, in a BSS, the AP will have a pairwise key for every STA and so must try to decrypt the packet with each of these keys, until it is successful. On average, an AP will need to attempt to decrypt the first 128-bit block with half of the total number of keys, which is half of the total number of STA, before it successfully decrypts the start of the packet.

This effect is amplified in an IBSS, where every STA must therefore hold keys for every other STA — effectively making every STA an AP in regards to WLS decryption cost.

Note that even under CBC mode, it is sufficient to attempt to decrypt just the first block in order to determine if the key is correct. The correct key can then continue to be applied to decrypting the remainder of the frame without interrupting the process.

The recvDATA() routine was modified to account for decrypting the first 128 bits of the MAC header with each of the available keys, until successful, to determine the destination address. This was achieved by multiplying the decryption cost for one 128-bit block by the sending node number. This simulates a STA decrypting the first 128 bits with each key for each node in sequence, until it reaches the key of the source node for this packet. On average, this will be half of all the active nodes in the simulation.

At the start of the simulation, with only one node transmitting (node_(1)), this will be the first key tried every time, but by the end of the simulation, with all of the nodes transmitting, this will be an overall average of half of the keys. The code was modified as shown here:

```
Mac802_11::recvDATA(Packet *p)
{
...
    /*
     * uptarget_->recv(p, (Handler*) 0);
     * Decrypting the maximum 2324-octet frame for Partial WLS
     * at 2324 * 8 / 1722000000 = 0.000010796748 seconds
     * which is far less than the MAC data rate 54Mb = 0.000344296
     * So, the only delay is encrypting the first 128 bit block
     * 128 / 1722000000 = 0.00000007433217 seconds
     * Decrypting the address will require multiple iterations
     * of decrypting the first 10 octets for each of the known "keys"
     * which will require the first 128 bits (one block) for each key
     * and will require min 1, max all keys tested
     * receiving from 1 STA = 0.00000007433217 seconds
     * receiving from 2 STA = 0.00000014866434 seconds
     * receiving from 5 STA = 0.00000037166086 seconds
     * receiving from 10 STA = 0.00000074332172 seconds
     * receiving from 25 STA = 0.00000185830430 seconds
     * receiving from 50 STA = 0.00000371660859 seconds
     * receiving from 100 STA = 0.00000743321718 seconds
     * receiving from 150 STA = 0.00001114982578 seconds
     * receiving from 254 STA = 0.00001888037166 seconds
     * struct hdr_mac802_11* mh = HDR_MAC802_11(p);
     * u_int32_t dst = ETHER_ADDR(mh->dh_ra);
     * u_int32_t src = ETHER_ADDR(mh->dh_ta);
     */
    // double addrdelay = 0.00000007433217;
    double addrdelay = src * 0.00000007433217;
    delayedPacket_ = p;
    Handler *postdecrypt_ = new DstDecTimer(this);
    // Scheduler::instance().schedule(postdecrypt_, delayedPacket_, 0.00000007433217);
    Scheduler::instance().schedule(postdecrypt_, delayedPacket_, addrdelay);
}
```

This code can be switched between PWLS and WLS by swapping the comments on the two `addrdelay` lines (0.00000007433217 seconds for PWLS and node-number*0.00000007433217 seconds for WLS).

The simulator was recompiled as PWLS and a new simulation of *drtests.tcl* based on *drtestg.tcl* was reconfigured in *drtestt.tcl*, as follows:

```
# PARTIAL WLS (56ns to encrypt, 74ns to decrypt first block)
# FULL WLS (56ns to encrypt, (74ns)*numbernodes to decrypt first block)
...
    set f_($i) [open drtest$t-PWLS$i.tr w]
...
    exec xgraph drtestg-node1.tr drtest$t-PWLS1.tr -geometry 800x400 &
```

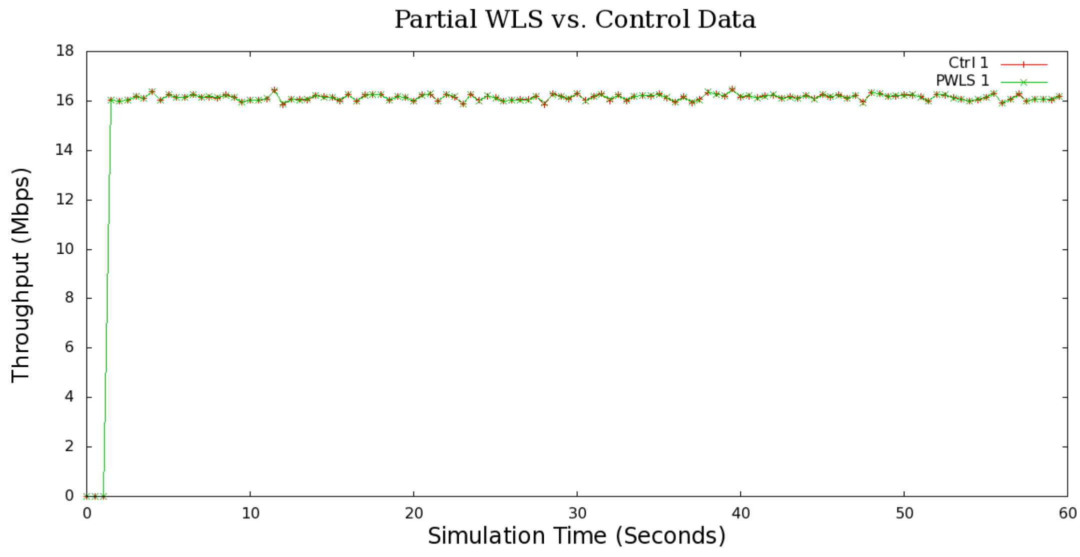


Figure 7.15: WLS Simulator configured for PWLS using 16.16 Mbps CBR

This simulation produced the same results as the previous *drtestr.tcl* and *drtests.tcl*, verifying the new implementation has not altered the PWLS results, as shown in Figure 7.15.

7.4.2 WLS Results

The *ns-2* code was then changed to simulate WLS and recompiled and the simulation was repeated in *drtestt.tcl*:

```
# PARTIAL WLS (56ns to encrypt, 74ns to decrypt first block)
# FULL WLS (56ns to encrypt, (74ns)*numbernodes to decrypt first block)
...
    set f_($i) [open drtest$t-WLS$i.tr w]
...
    exec xgraph drtestg-node1.tr drtest$t-PWLS1.tr drtest$t-WLS1.tr
-geometry 800x400 &
```

This simulation produced the same results as the previous simulations, with throughput identical to PWLS, following the original control data in Figure 7.1.

These results showed that, using reasonable specifications, WLS also has no effect on throughput in this two-node scenario, as shown in Figure 7.16.

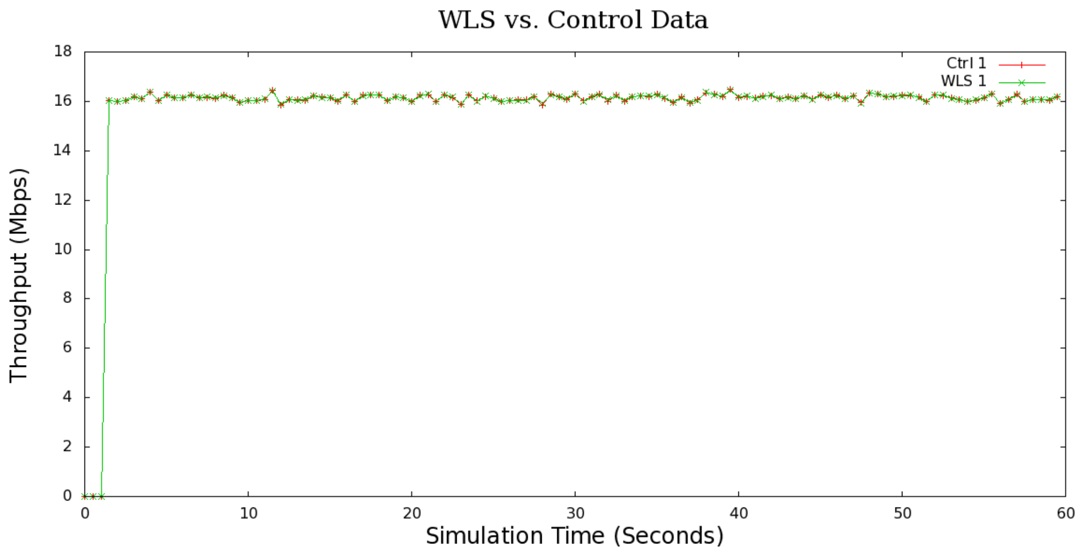


Figure 7.16: WLS Simulator configured for WLS using 16.16 Mbps CBR

7.5 Complete PWLS and WLS Results

With the implementation developed, the code was then changed back to simulate PWLS and recompiled and all of the control simulations were repeated under the new implementation to enable a thorough analysis of the effects of these proposals. Then, the code was switched to simulate WLS, recompiled and again all of the control simulations were repeated under the WLS implementation to compare the effects of these proposals.

These are presented here, as pairs of PWLS and WLS results, to enable easy comparison, however all the PWLS results were in fact gathered before any the WLS results, as switching between the two requires recompiling the *ns-2* simulator.

7.5.1 Two CBR Transmitters

The control simulation of *drtesth.tcl* was reconfigured from *drtestt.tcl*, as *drtestuh.tcl* and subsequently as *drtestvh.tcl*, as follows:

```
set val(nn)        3                        ;# number of mobilenodes
set val(sc)        "drtest-3-2-scen-stat"   ;# scene movement file
set val(cp)        ""                       ;# traffic pattern file
```

The scene movement file, *drtest-3-2-scen-stat*, has two sending nodes 10 m either side of the central `node_(0)` receiver, for the entire simulation, with no movement and remaining within the 250 m radio range throughout the simulation. Thus, the receiver can “hear” both sending nodes and each node can hear both the receiver and the other node.

This produced effectively the same results as the control case, as shown in Figures 7.17 and 7.18. Note these graphics plot the data with lines only between the measured data for clarity — the data points occur every 0.5 seconds (at the vertices) on each of the lines.

The control simulation of *drtesti.tcl* was reconfigured from *drtestuh.tcl*, as *drtestui.tcl* and subsequently as *drtestvi.tcl*, as follows:

```
set val(sc)        "drtest-3-2-scen-dist"   ;# scene movement file
```

The scene movement file, *drtest-3-2-scen-dist*, has two sending nodes 150 m either side of the `node_(0)` receiver, with no movement throughout the simulation. Thus, the receiver can hear both sending nodes and each node can hear the

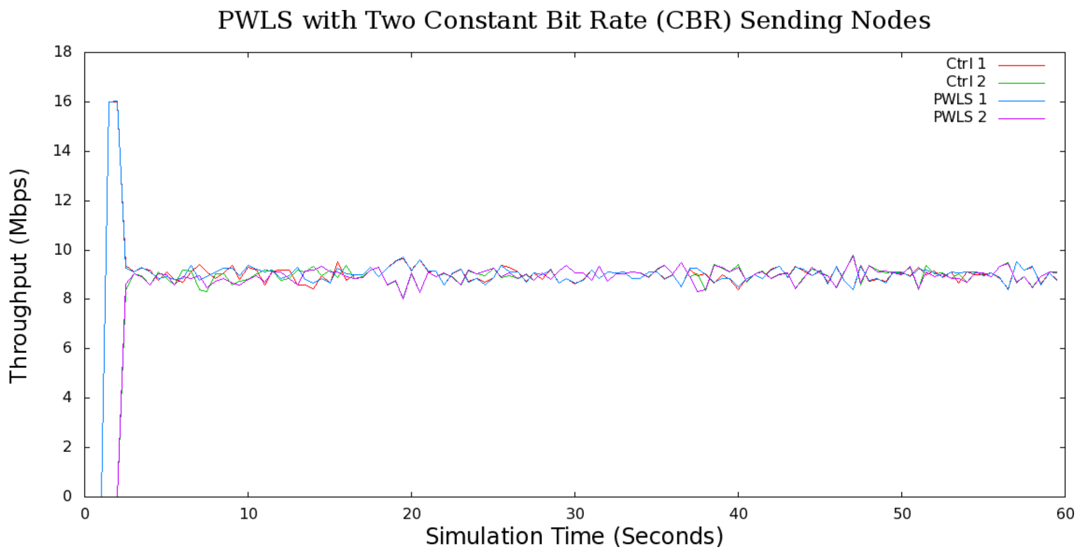


Figure 7.17: PWLS with two CBR nodes 10 m opposite equidistant from AP

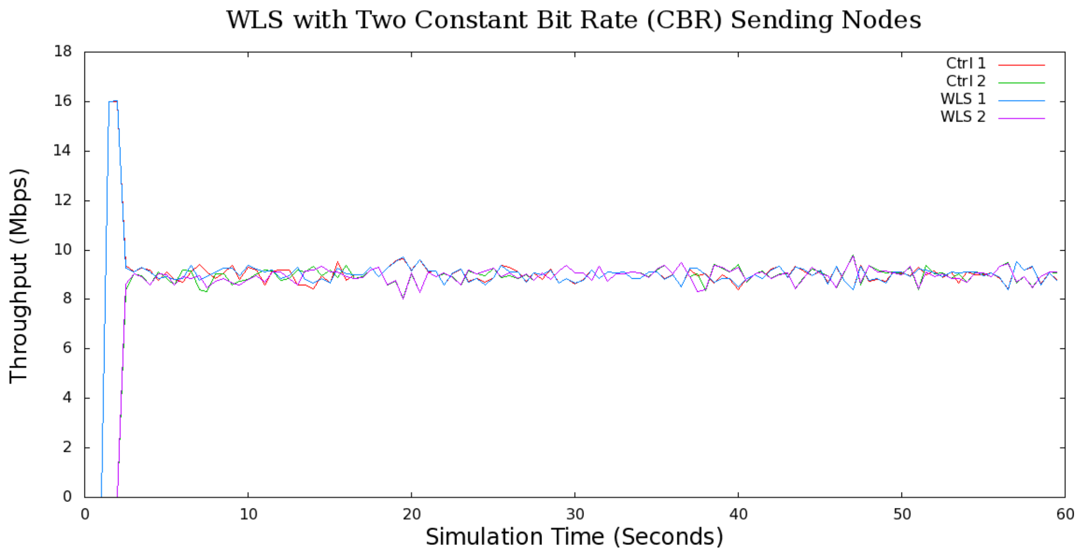


Figure 7.18: WLS with two CBR nodes 10 m opposite equidistant from AP

receiver but can not hear the other sending node. This is the so-called "hidden node" case.

Again, this produced effectively the same results as the control case, as shown in Figures 7.19 and 7.20.

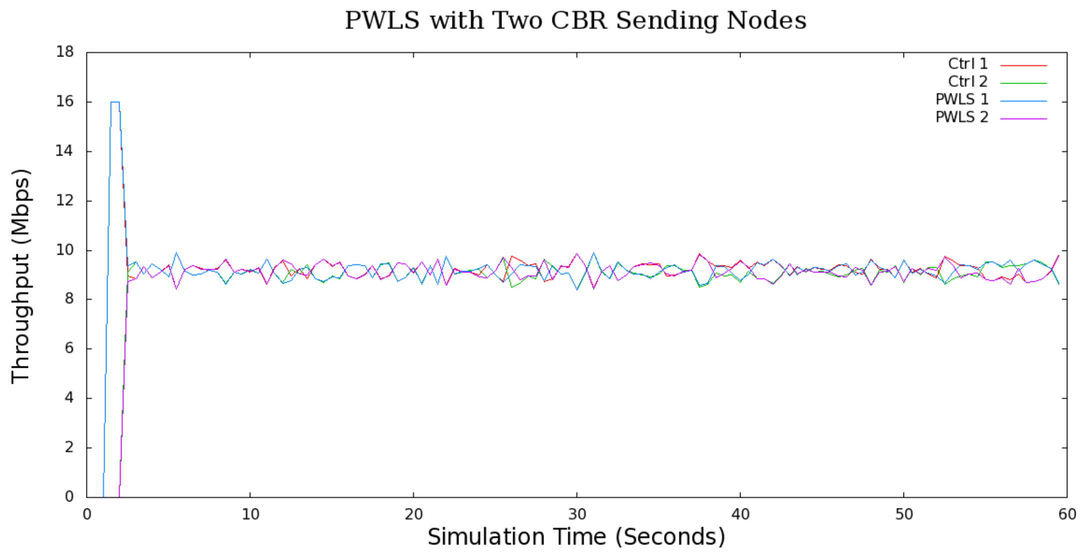


Figure 7.19: PWLS with two CBR nodes 150 m opposite equidistant from AP

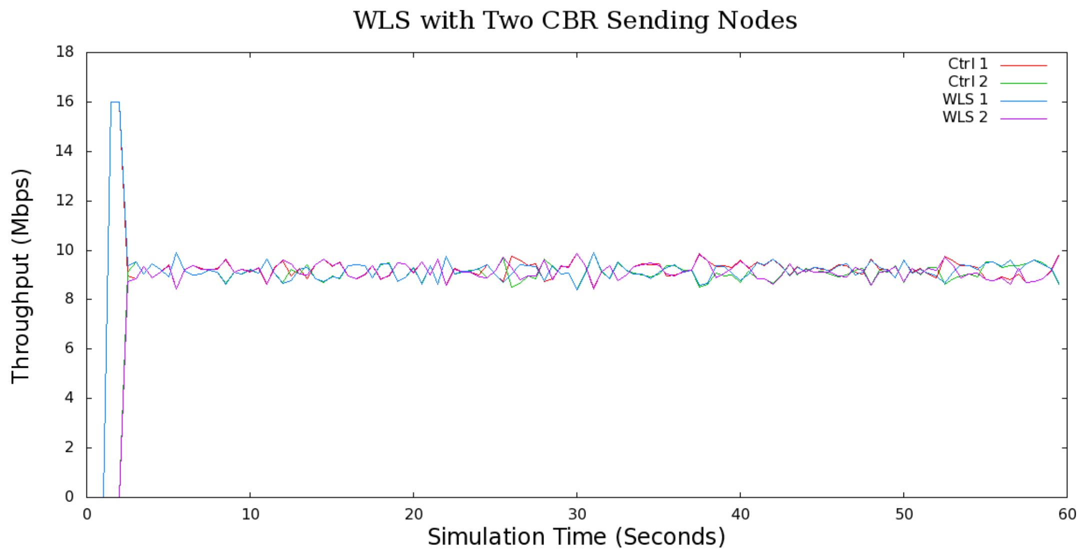


Figure 7.20: WLS with two CBR nodes 150 m opposite equidistant from AP

The next case to be tested was where the nodes were not equidistant from the receiver. Thus, the receiver can hear both sending nodes and each node can hear the receiver but can not hear the other sending node — hidden nodes — and each sender has a different propagation delay to the receiver. The relevant settings were configured in the scene movement file, *drtest-3-2-scen-diff*, configured from *drtestuj.tcl*, as *drtestuj.tcl* and subsequently as *drtestvj.tcl*.

Once again, this produced effectively the same results as the control case, as shown in Figures 7.21 and 7.22.

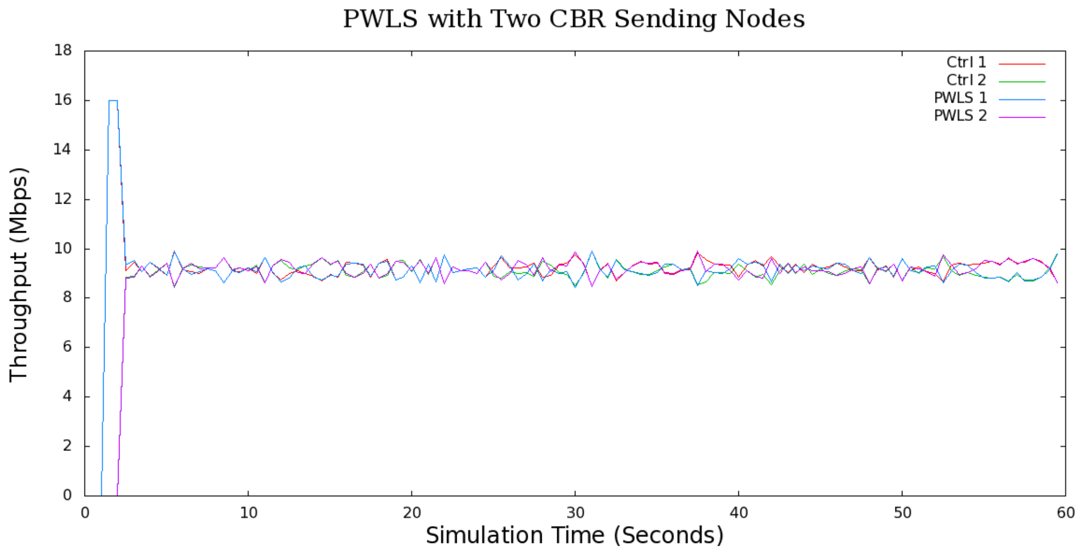


Figure 7.21: PWLS with two CBR nodes at different distances from the AP

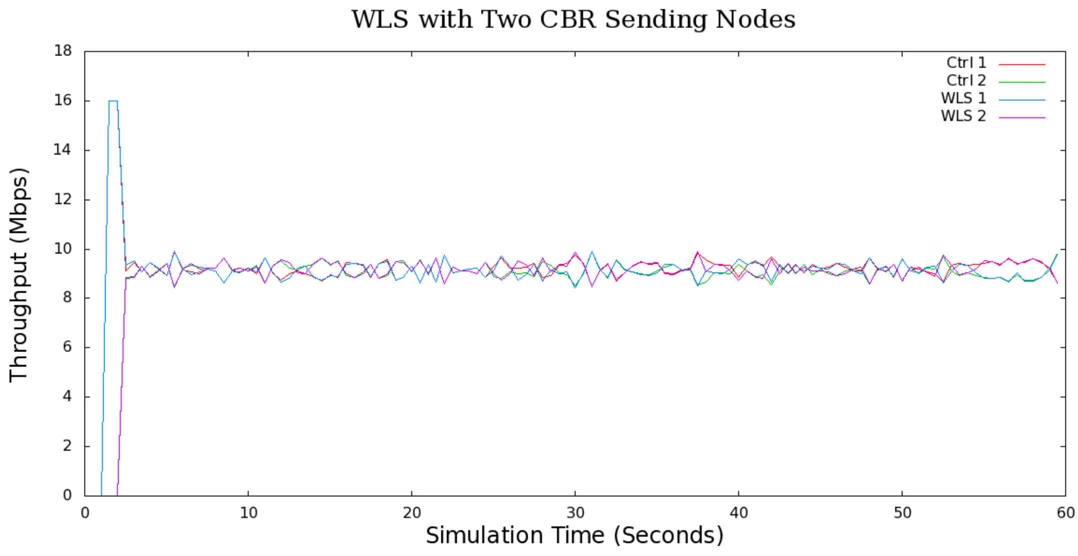


Figure 7.22: WLS with two CBR nodes at different distances from the AP

The next simulation recorded the average throughput achieved by any one sending STA (out of two, in this case) in a single trace file. This was achieved by using the same scenario, including the same scene movement file as the previous simulation and modifying the `proc record {}` to sum all the data received by all the sinks, divide by the number of active nodes and record the results in a single trace file `f0`, *drtestuk-avgbw-2nPWLS.tr* and *drtestvk-avgbw-2nWLS.tr*, as previously described.

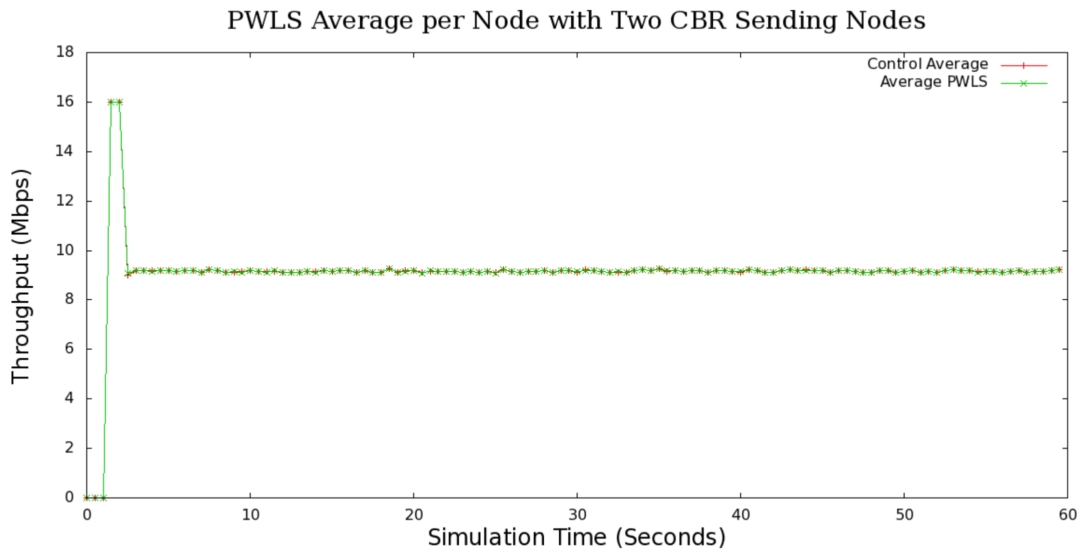


Figure 7.23: Average PWLS throughput with two CBR nodes

7.5.2 Ten CBR Transmitters

The next step was to test ten high-output STAs in contention for the same channel.

The same random 11-node scene movement file, *drtest-11-10-scen*, was used in *drtestul.tcl* and subsequently *drtestvl.tcl*, with CBR generators attached to UDP agents on `node_(1)` through `node_(10)`, each starting at time $5.0 * nodeindex$, to give five second intervals between increasing numbers of sending nodes; and all the sinks on `node_(0)`.

Once more, this produced effectively the same results as the control case, as shown in Figures 7.25 through 7.32.

Even the highly-timing-sensitive individual node traces follow the same trend as the control data. The individual traces for `node_(1)` under both PWLS and

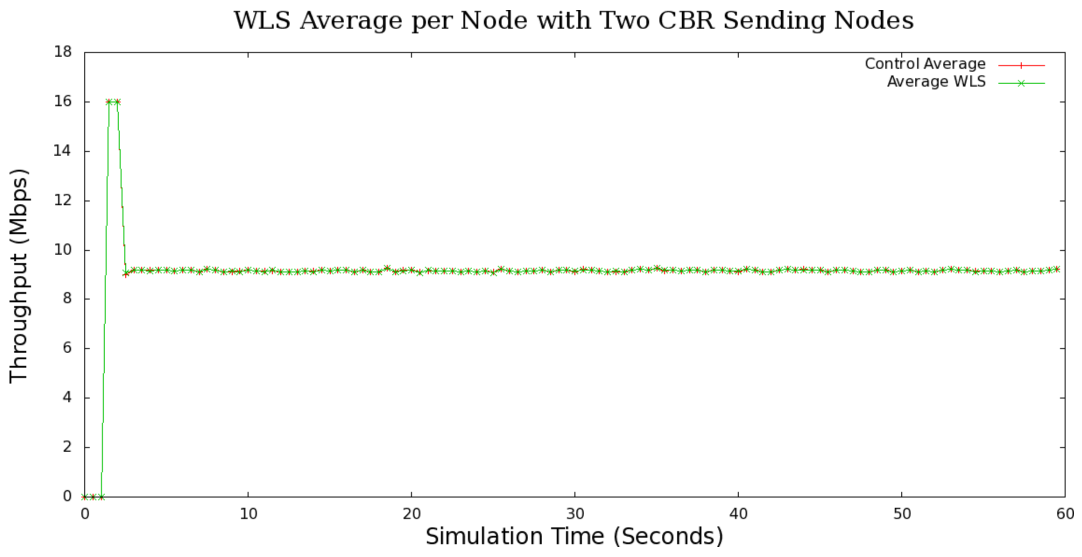


Figure 7.24: Average WLS throughput with two CBR nodes

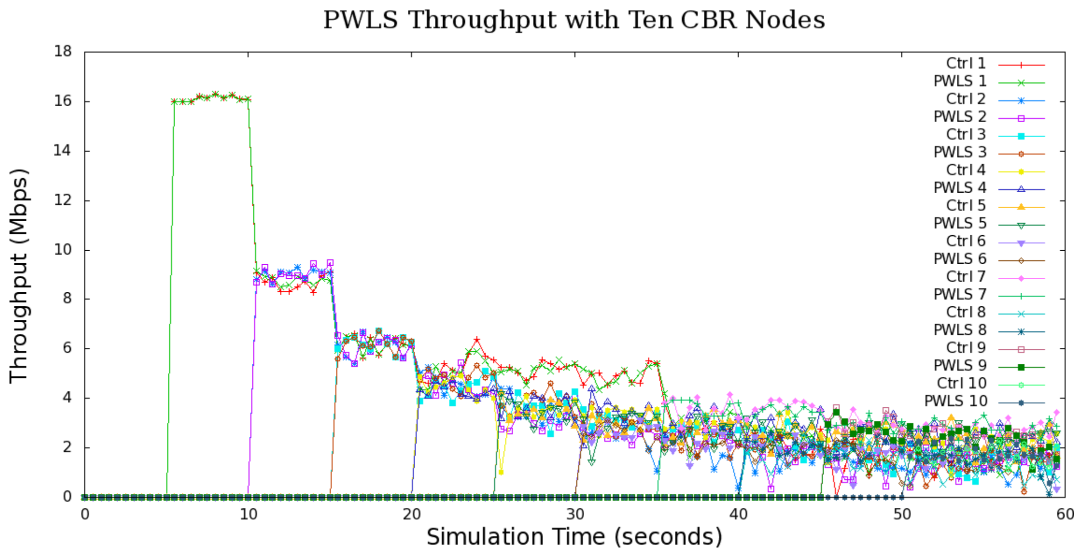


Figure 7.25: PWLS throughput with ten CBR nodes

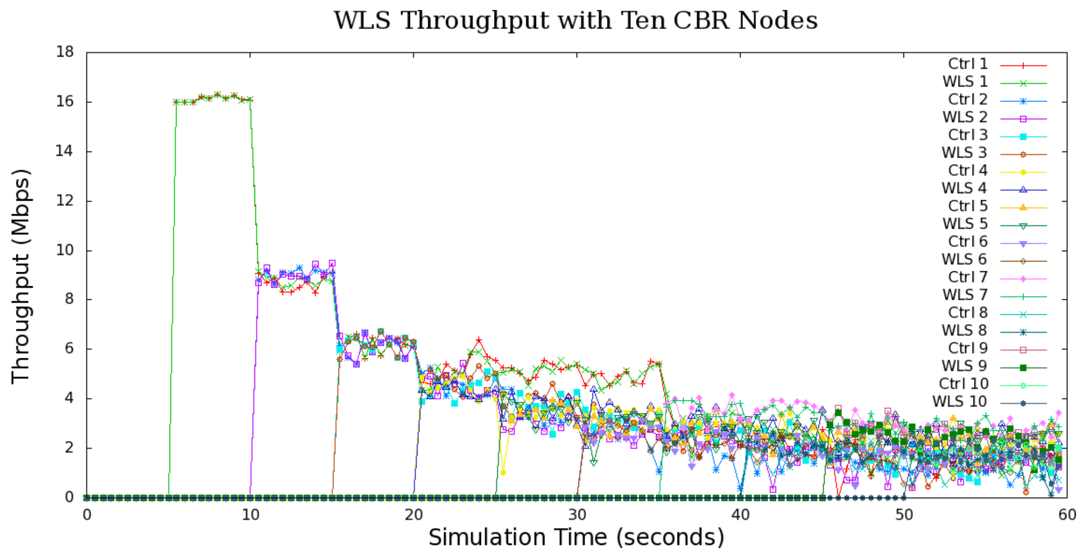


Figure 7.26: WLS throughput with ten CBR nodes

WLS are plotted over the control trace in Figures 7.27 and 7.28.

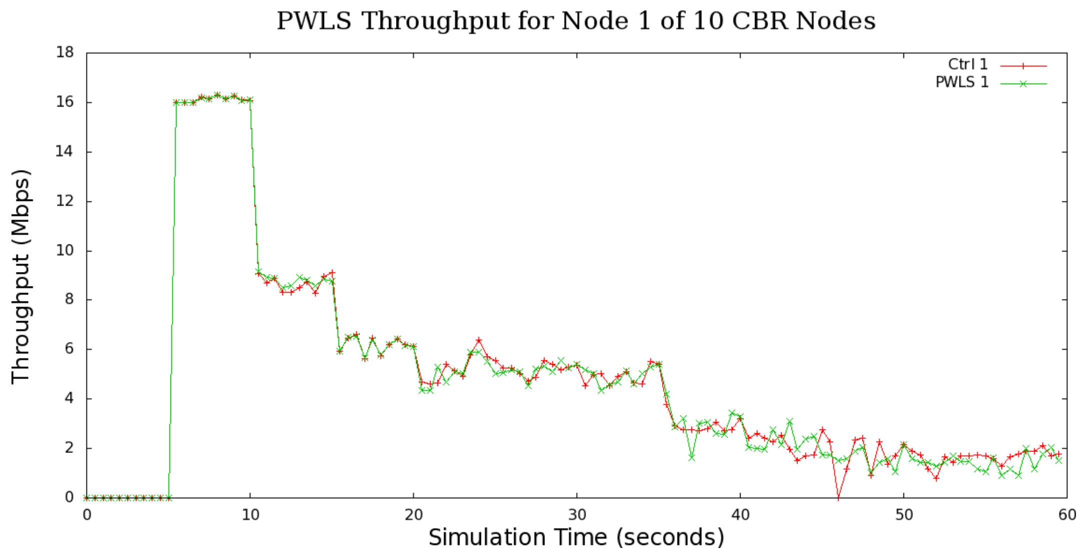


Figure 7.27: Node_(1) PWLS throughput with ten CBR nodes

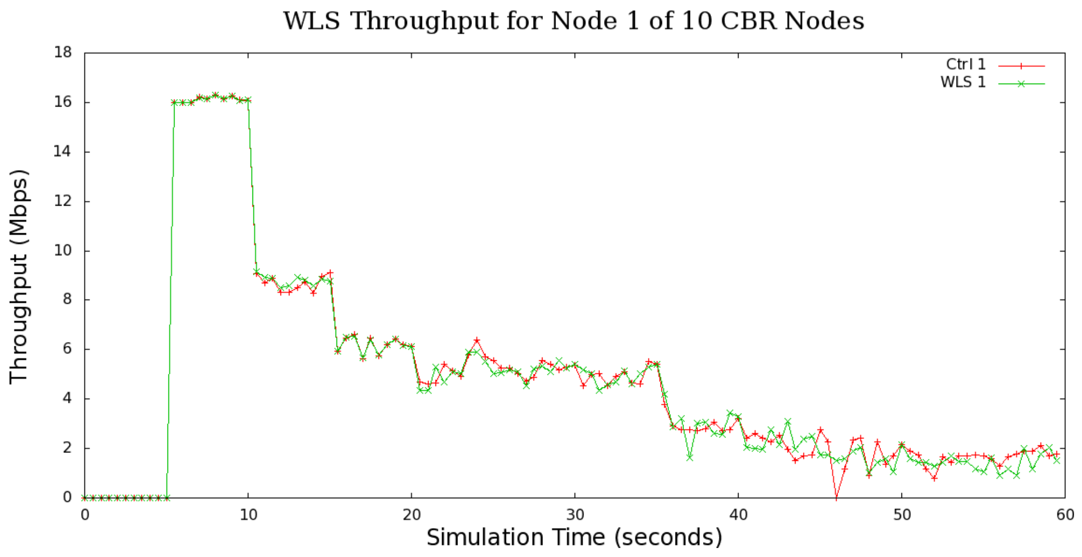


Figure 7.28: Node_(1) WLS throughput with ten CBR nodes

The individual traces for `node_(5)` under both PWLS and WLS are plotted over the control trace in Figures 7.29 and 7.30.

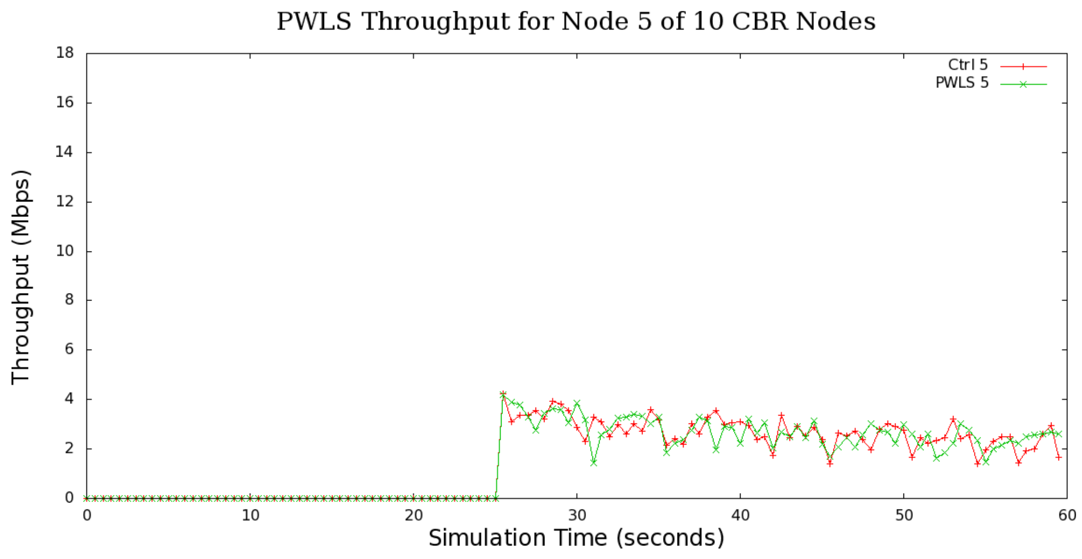


Figure 7.29: Node_(5) PWLS throughput with ten CBR nodes

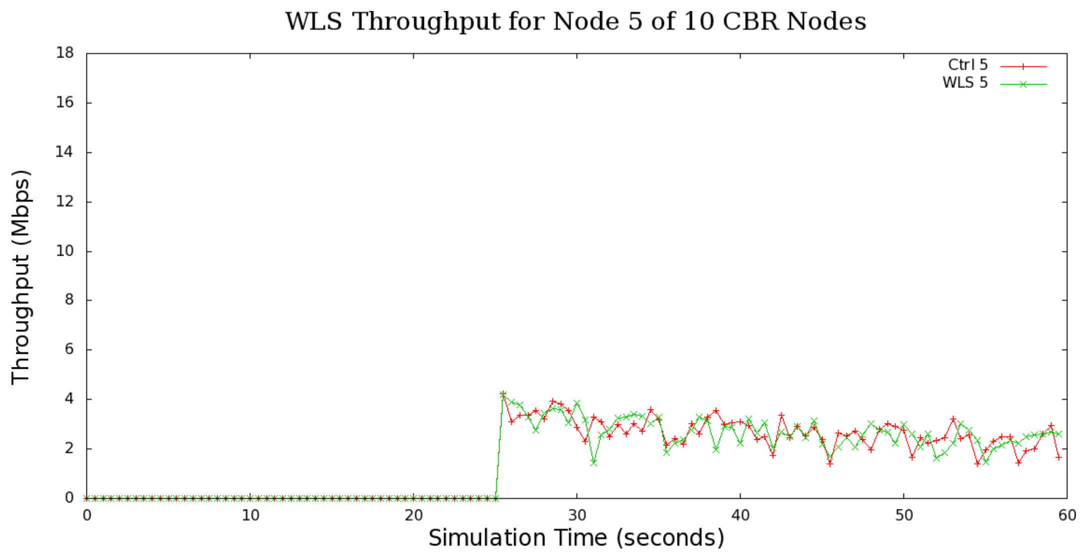


Figure 7.30: Node_(5) WLS throughput with ten CBR nodes

The individual traces for `node_(10)` under both PWLS and WLS are plotted over the control trace in Figures 7.31 and 7.32.

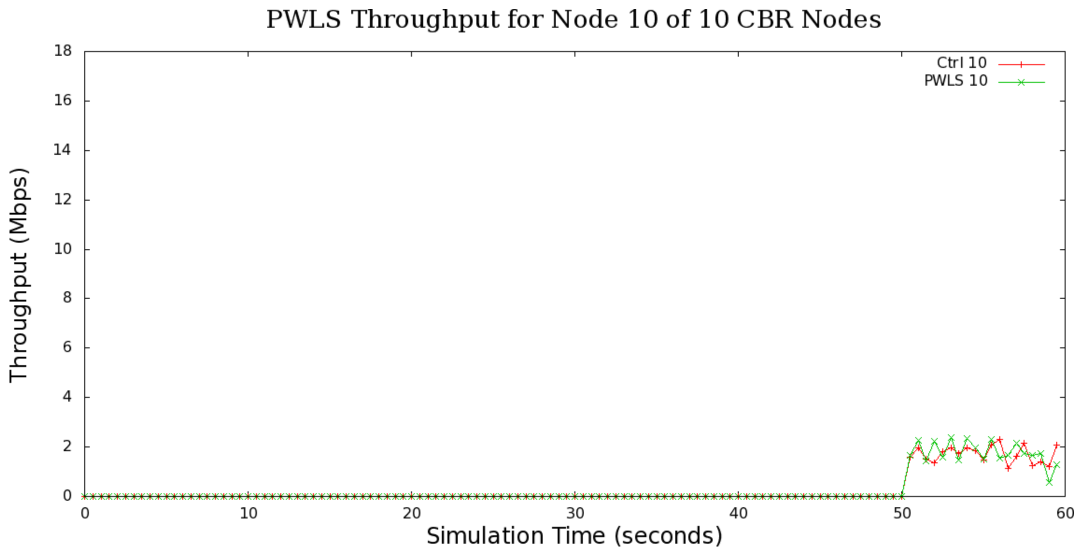


Figure 7.31: Node_(10) WLS throughput with ten CBR nodes

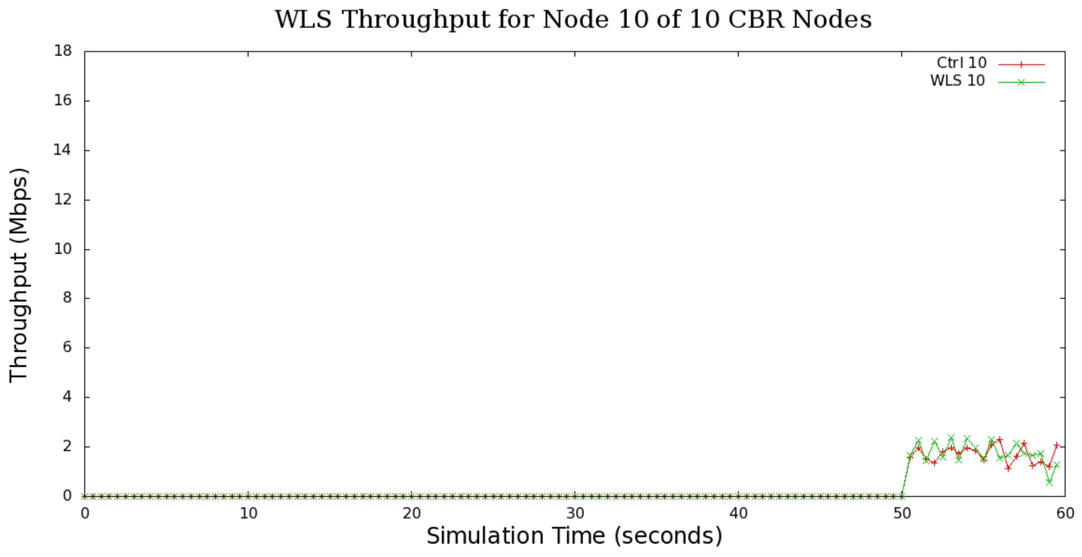


Figure 7.32: Node_(10) WLS throughput with ten CBR nodes

The average throughput per active node under both PWLS and WLS are plotted over the control trace in Figures 7.33 and 7.34.

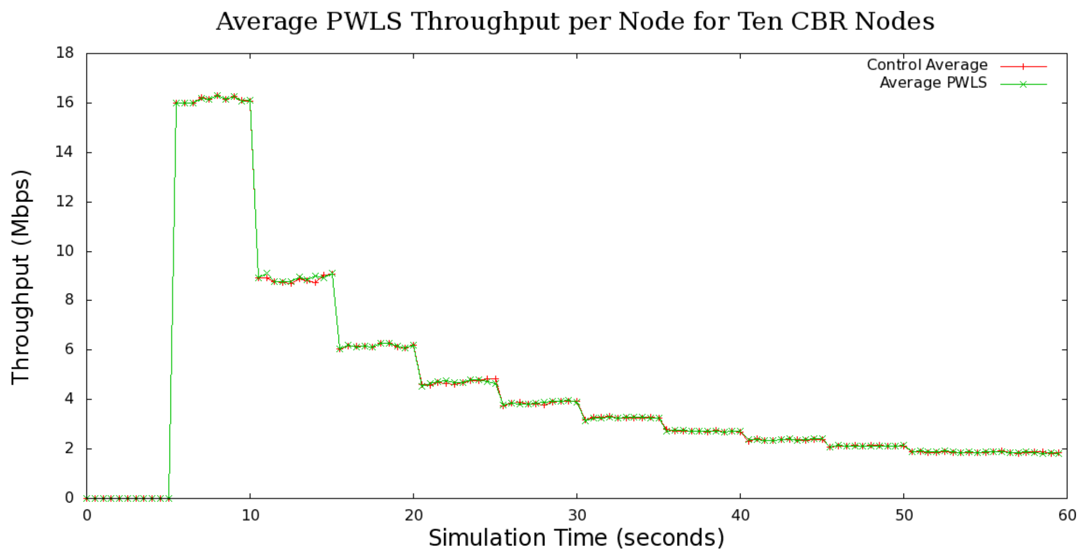


Figure 7.33: Average PWLS throughput with ten CBR nodes

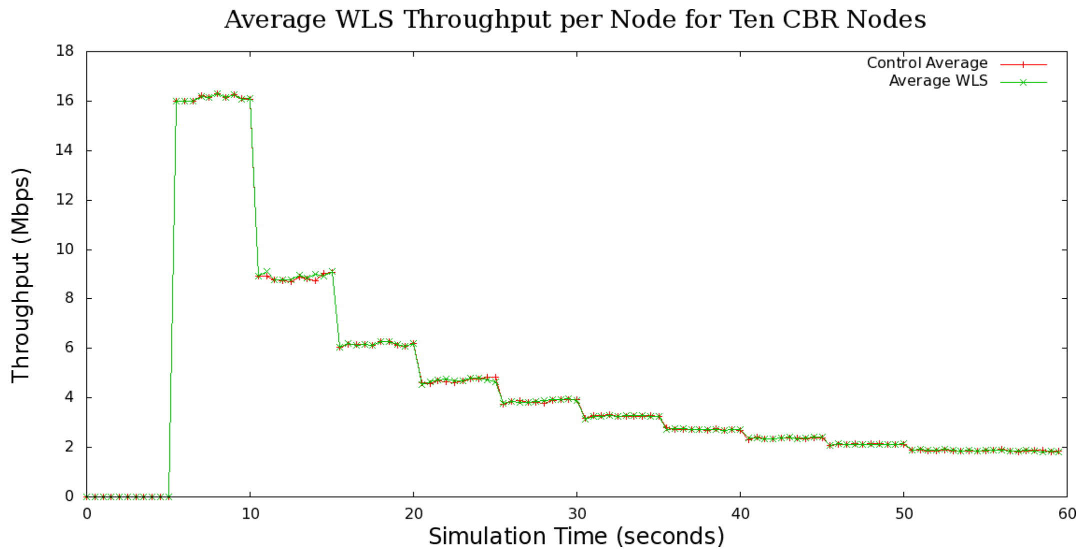


Figure 7.34: Average WLS throughput with ten CBR nodes

7.5.3 TCP Traffic

The scenario of *drtestn.tcl* was repeated in *drtestun.tcl* and subsequently *drtestvn.tcl*, to create and start FTP sources transferring a 2 MB file on TCP agents to and from alternate nodes and `node_(0)`.

Again, this produced effectively the same results as the control case, as shown in Figures 7.35 and 7.36.

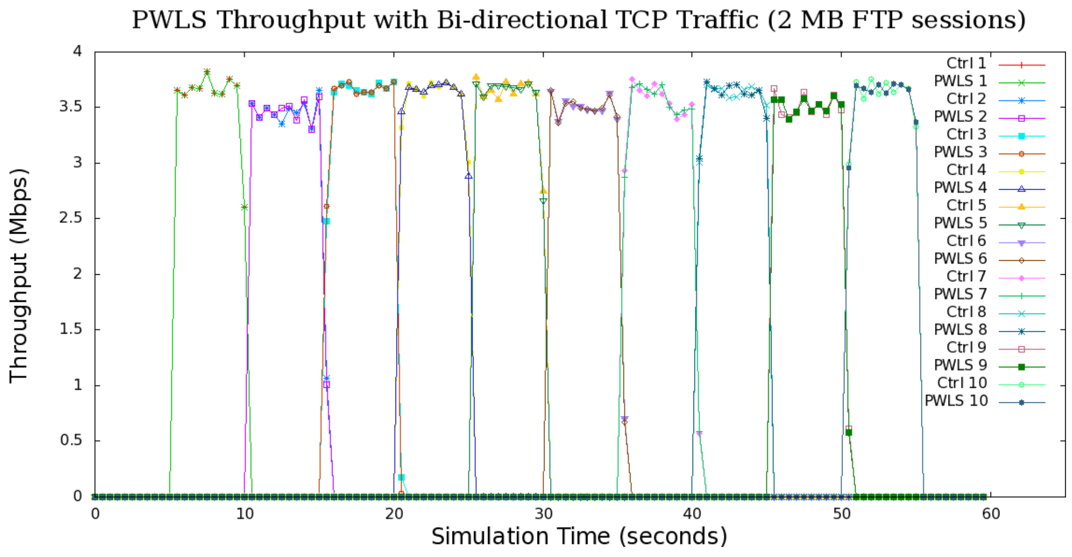


Figure 7.35: PWLS throughput with 2 MB FTP Transfers

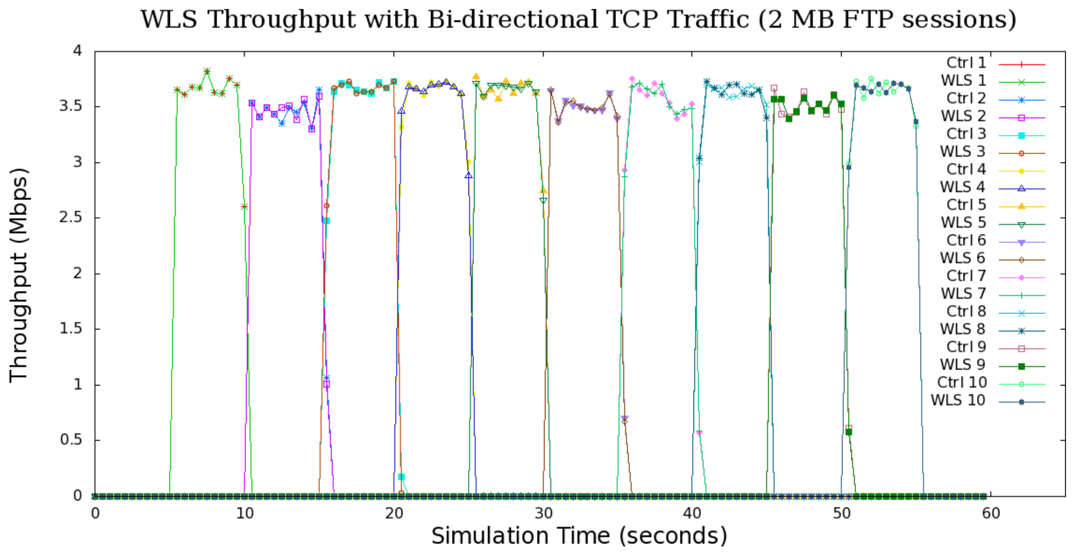


Figure 7.36: WLS throughput with 2 MB FTP Transfers

The total TCP throughput for all concurrent transfers under both PWLS and WLS are plotted over the control trace in Figures 7.37 and 7.38.

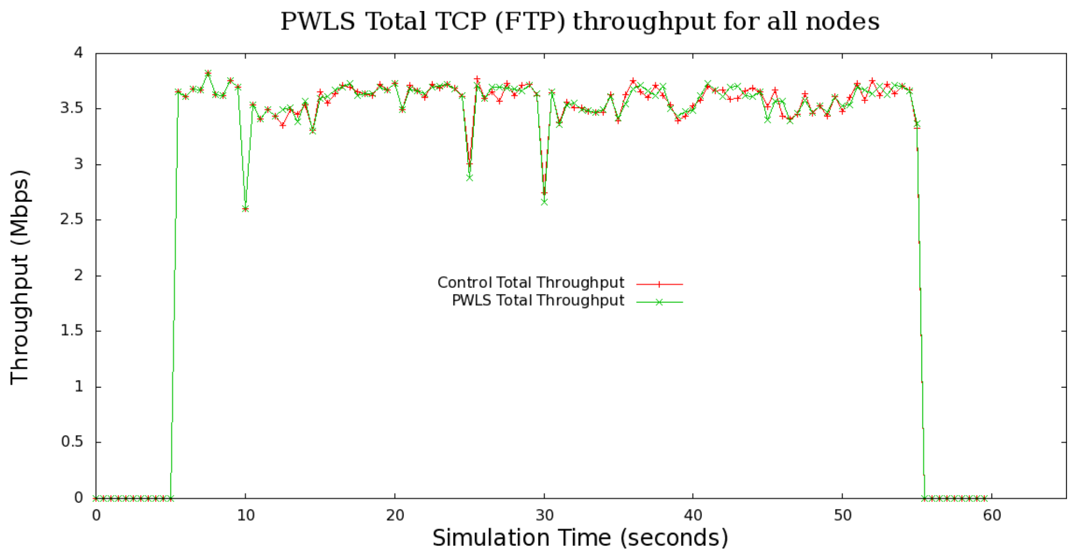


Figure 7.37: Average PWLS throughput with 2 MB FTP Transfers

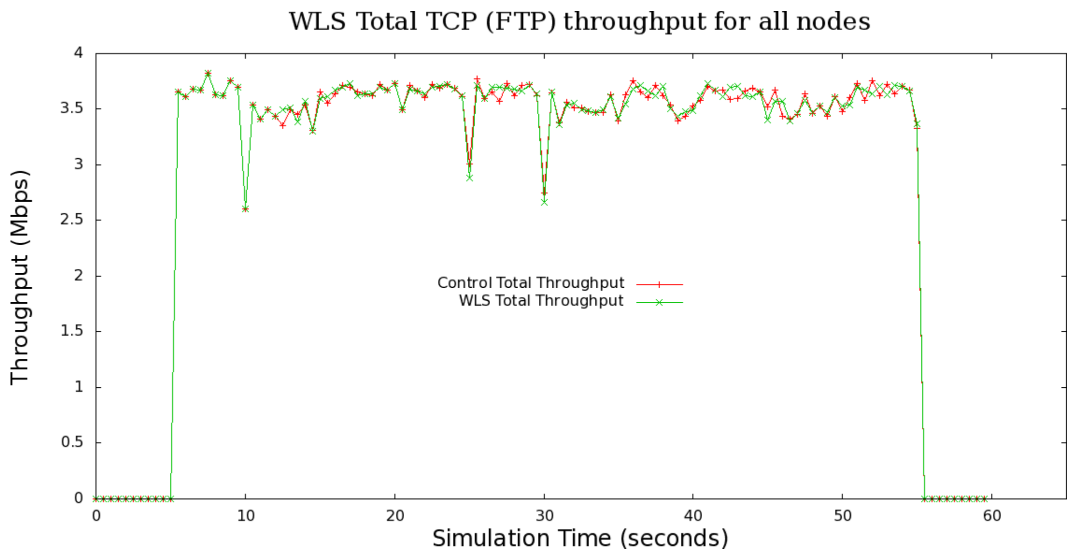


Figure 7.38: Average WLS throughput with 2 MB FTP Transfers

7.5.4 Overlapping TCP Traffic

The scenario of *drtestp.tcl* was repeated in *drtestup.tcl* and subsequently *drtestvp.tcl*, to create and start FTP sources transferring a 2.6 MB file on TCP agents to and from alternate nodes and `node_(0)`.

Once again, this produced effectively the same results as the control case, as shown in Figures 7.39 through 7.46.

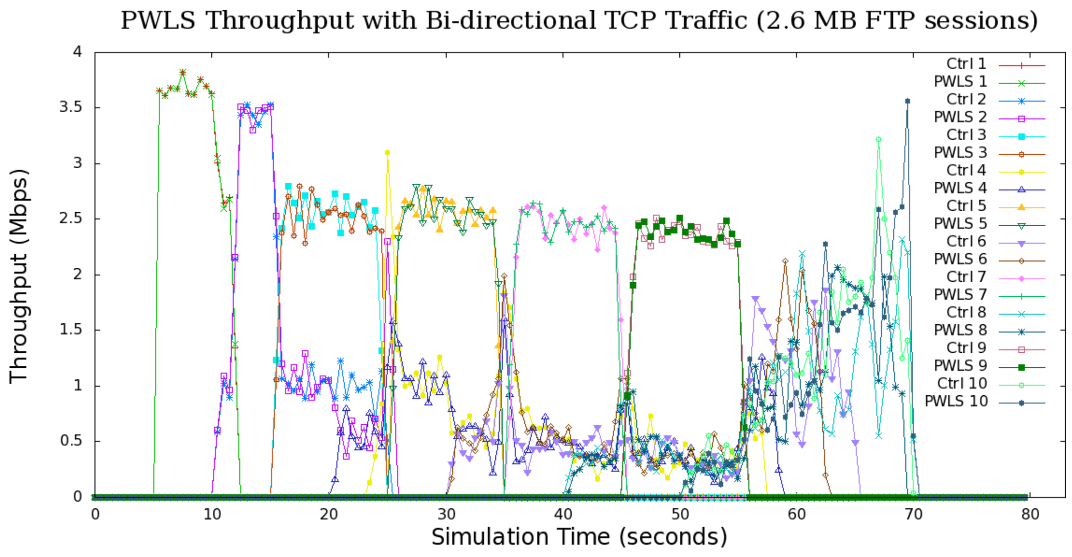


Figure 7.39: PWLS throughput with 2.6 MB overlapping FTP Transfers

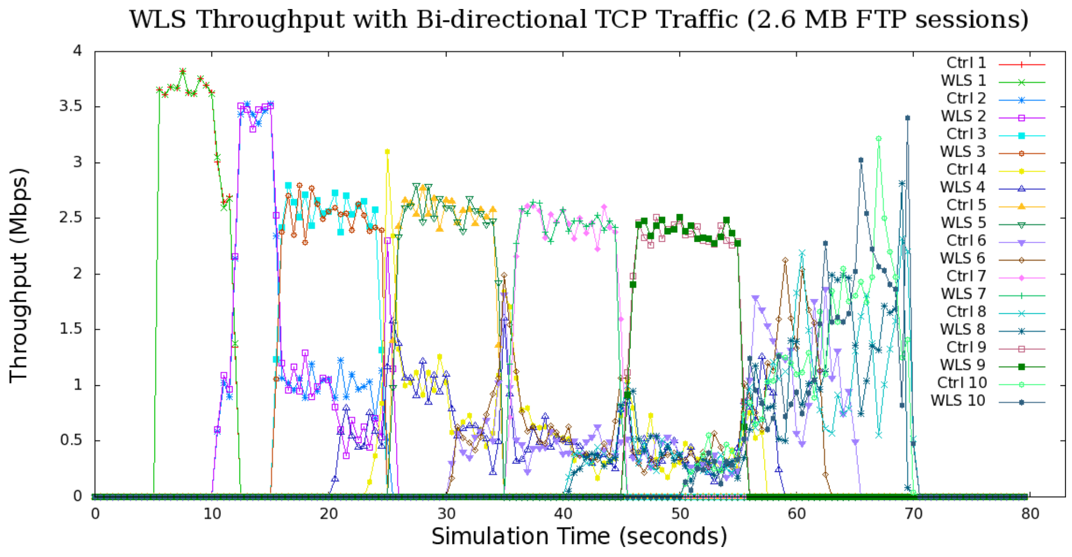


Figure 7.40: WLS throughput with 2.6 MB overlapping FTP Transfers

The individual traces for `node_(1)` under both PWLS and WLS are plotted over the control trace in Figures 7.41 and 7.42.

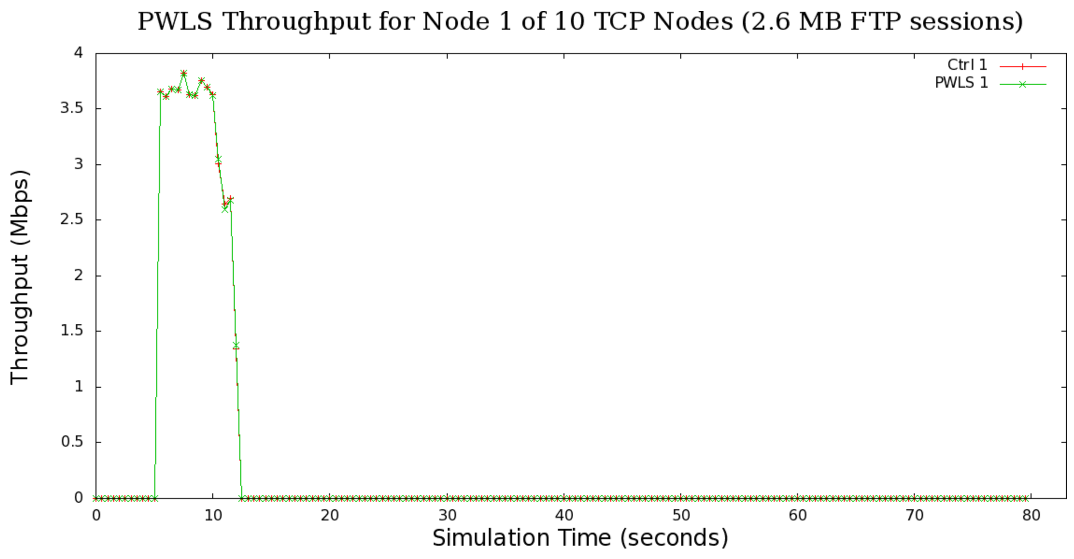


Figure 7.41: Node_(1) PWLS throughput with 2.6 MB FTP Transfers

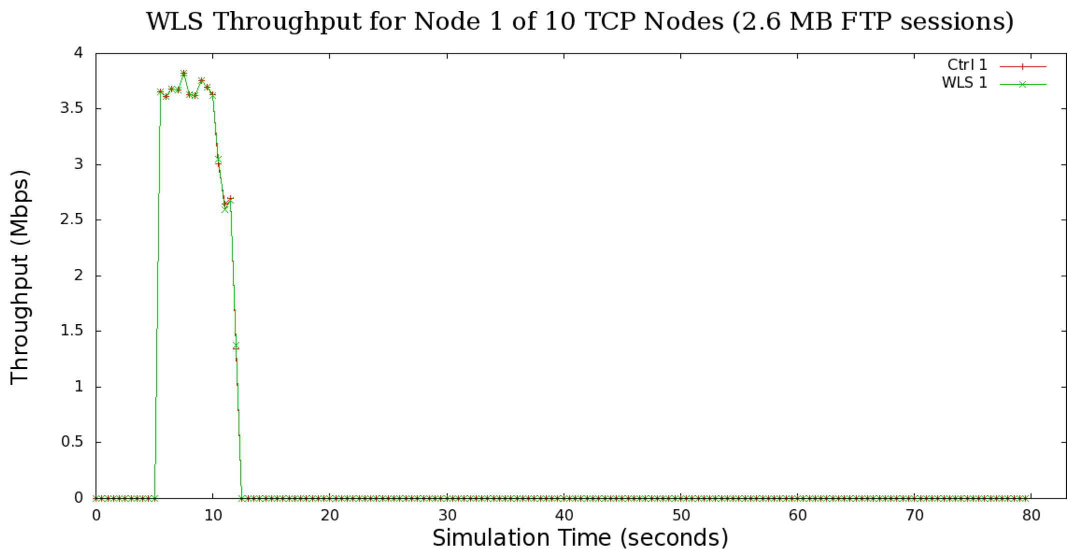


Figure 7.42: Node_(1) WLS throughput with 2.6 MB FTP Transfers

The individual traces for `node_(5)` under both PWLS and WLS are plotted over the control trace in Figures 7.43 and 7.44.

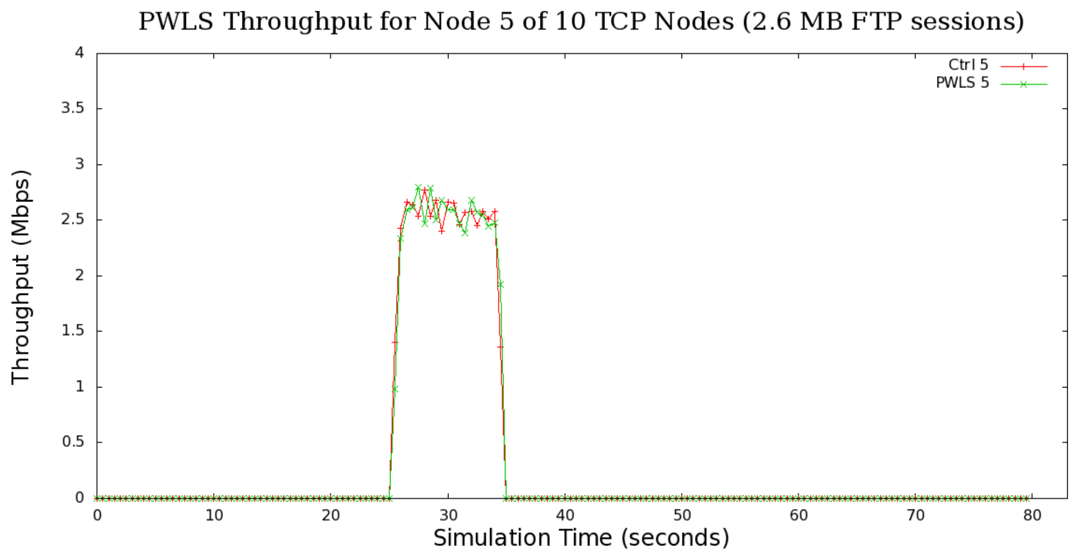


Figure 7.43: Node_(5) PWLS throughput with 2.6 MB FTP Transfers

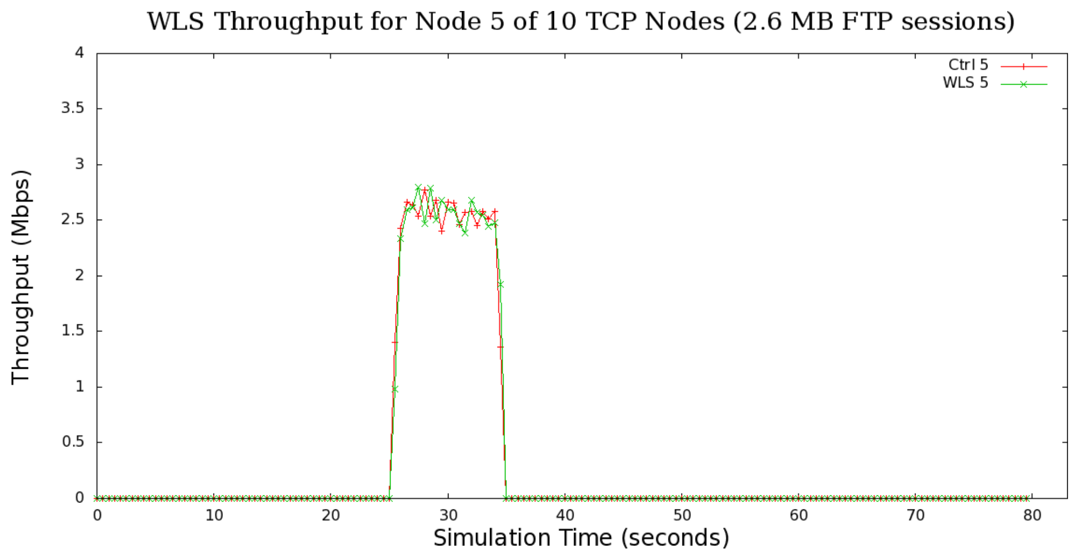


Figure 7.44: Node_(5) WLS throughput with 2.6 MB FTP Transfers

The individual traces for `node_(10)` under both PWLS and WLS are plotted over the control trace in Figures 7.45 and 7.46.

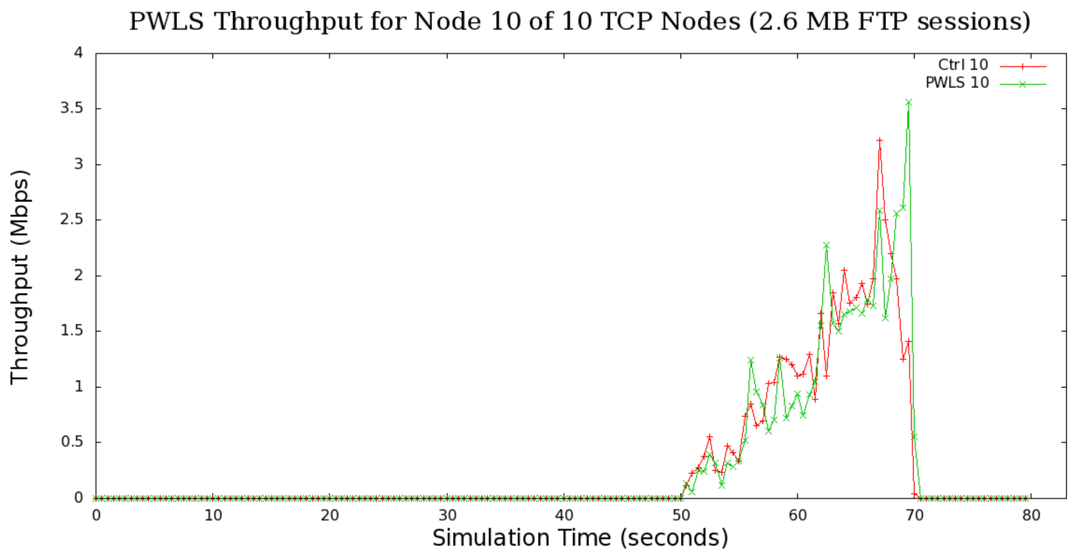


Figure 7.45: Node_(10) PWLS throughput with 2.6 MB FTP Transfers

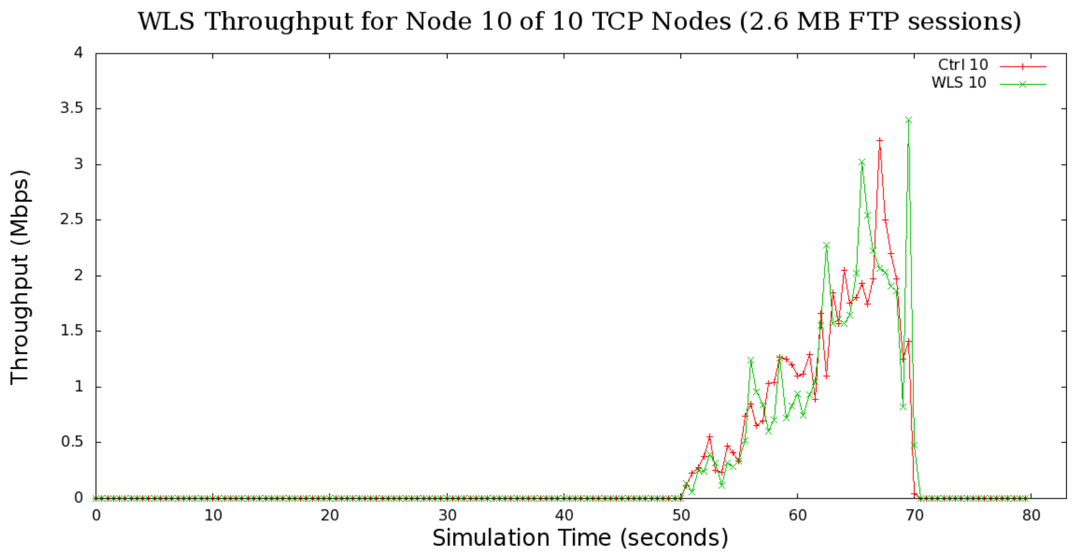


Figure 7.46: Node_(10) WLS throughput with 2.6 MB FTP Transfers

The total TCP throughput for all concurrent transfers under both PWLS and WLS are plotted over the control trace in Figures 7.47 and 7.48.

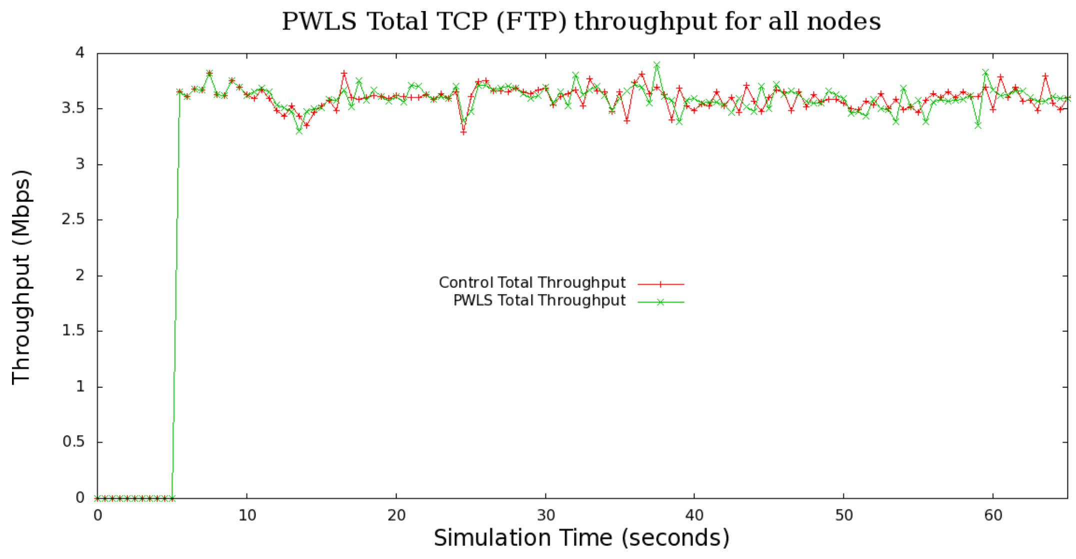


Figure 7.47: Total PWLS throughput with 2.6 MB FTP Transfers

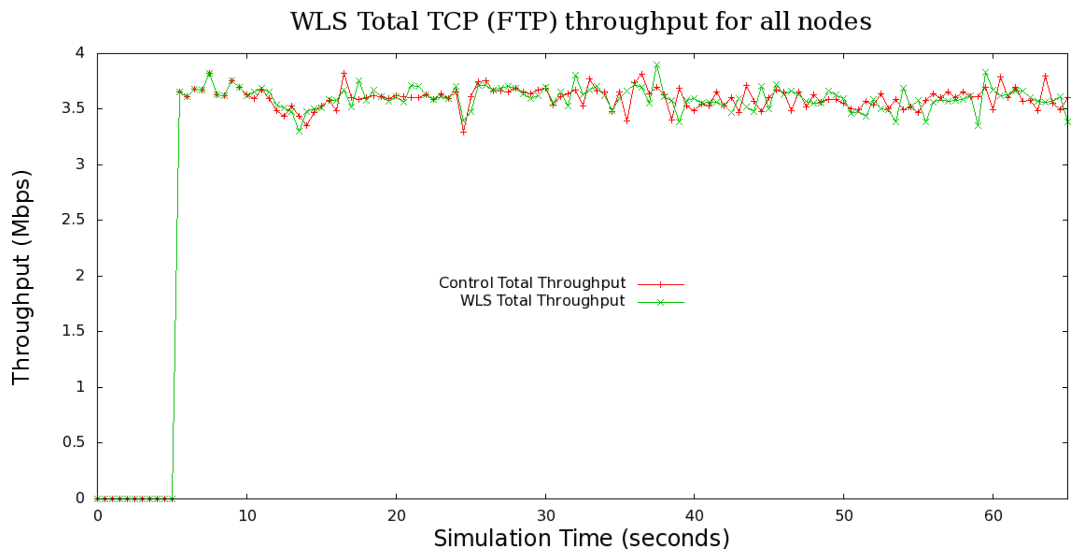


Figure 7.48: Total WLS throughput with 2.6 MB FTP Transfers

7.6 Discussion

These results conclusively show that reasonable rates of encryption can be expected to have no effect on the network throughput for either partial WLS or full WLS, as proposed here. This is due to the already extensive wait times for the base protocol requiring positive acknowledgement of all directed traffic. The delays introduced by the need for positive acknowledgements, combined with the collision avoidance protocol, more than cover any reasonable delays caused by additional encryption at this late stage.

However, the results show that should the encryption rate be so slow, as to delay packet transmission by values approaching two microseconds, this then has a catastrophic effect on network throughput. This is due to the transmission timeouts allowing for a maximum medium propagation delay of two microseconds. As such, encryption delays approaching two microseconds trigger the timeouts, causing retransmission (retransmission should not include an encryption overhead) however retransmission does involve increasing the contention window with a new contention backoff for every new packet.

This has a devastating effect on network throughput, dramatically reducing the throughput, causing failures on retransmission for all directed packets.

Normally the encryption rate will be very much faster than the data rate. As such, it was assumed that the relevant delay in this case was the encryption of the first ciphertext unit, which must occur before transmission can begin. Assuming a worst case of a 128-bit block cipher in Cipher-Block-Chaining (CBC) mode, this would require the first 128-bit encryption to be completed before transmission begins.

However, this is not actually true. The PLCP Preamble will be transmitted first at 1 Mbps. This is a fixed sync pattern and a fixed SOF delimiter, which should not be encrypted, as it would provide one or more blocks suitable for a known-plaintext attack on the encryption and, irrespective of this, must not be encrypted as a technical requirement to synchronise the receiving radio. A standard PLCP Preamble will take 144 µs to transmit, followed by the 48-bit PLCP header, also at 1 Mbps — a total of 192 µs. The short PLCP Preamble will take 72 µs to transmit, followed by the 48-bit PLCP header, now at 2 Mbps — a total of 96 µs.

This gives an additional window to buffer greater cryptographic delays. Combining this with the possibly of adjusting the transmission timeouts to account for any encryption (and receiver decryption) delays, provide a further enhancement detailed in Chapter 8.

Chapter 8

Analysis and Simulation Enhancement

"It is a bad plan that admits of no modification." — Publilius Syrus, *Maxims*, ~100 BC.

From the previous results, this chapter details further modifications to enhance the simulations and draws fresh results from these new simulations, which are then analysed here.

8.1 General Outline

The results of the previous Chapter 7 provide valuable insight into the likely practical operation of a WLAN utilising WLS and also highlight a number of deficiencies in the original assumptions and the configuration of the simulations.

These results show, for any reasonable rates of encryption and decryption, neither PWLS nor WLS would have any negative impact on a network throughput. However, these results provide only limited information on the effects of very slow rates of either or both encryption or decryption, inducing considerably longer delays into the system.

The results have shown that until the encryption penalties have induced sufficient delay to cause retransmission timeouts to be triggered, adding low level encryption *has no effect* on network performance. This is due to the existing delays in the WLAN protocols necessary for multiple access with collision avoidance. These inbuilt delays mask any reasonable cryptographic penalties introduced by this proposal.

However, with the dramatic cut off as the retransmit timers expired, these results provided little information on the actual characteristics of the encryption penalty past these limits, if the timeouts were to be adjusted to allow for the additional encryption delays.

It was decided to enhance the simulator environment, to increase the retransmission timeouts in line with any increased encryption delays. This was achieved by including calculations for decryption times during the encryption process and adding any excess decryption time to the Send timeout.

8.2 Refining the Investigation

Additional enhancements included at this stage included making an additional allowance for the change in packet size due to the padding while encrypting within a 128 bit of block cipher; adding the decryption delay to broadcast packets as well as directed traffic; and adding an allowance for an additional decryption attempt by every STA, to first check the broadcast key before its unicast keys.

These were achieved by modifying the `Mac802_11` class *C++* source code, in the file *~ns2/mac/mac-802_11.cc*, as described below.

First the simulator code used to produce the previous results was saved:

```
cd /usr/src/ns-2.33
cd mac
ls -al
cp -p mac-802_11.h mac-802_11.h.dr1
cp -p mac-802_11.cc mac-802_11.cc.dr1
ls -al
vi mac-802_11.cc
```

8.2.1 Variable Encryption Delay

The `Mac802_11` class is a subclass of the `Mac` class, itself a subclass of the `BiConnector` subclass of the `NsObject` class. As such, the `Mac802_11` class inherits the `double delay_; // MAC overhead` from the `Mac` class, but which is unused by the default *ns-2.33* `Mac802_11` class.

The author utilised this variable to provide an efficient method to vary the cryptographic penalties without requiring the entire recompilation of the *ns-2* simulator every time.

In the authors modifications, the `Mac802_11 delay_` is used to hold the *decryption* delay for 128 bits. The decryption delay is used to minimise error as decryption is usually slower than encryption and hence the decryption time is usually the larger value.

Internally the routines use a ratio of 1.722:2.257 encryption delay to decryption delay, based on Sivakumar and Velmurugan's CCMP AES Cipher for WLAN, providing throughputs of 2.257 Gbps for encryption and 1.722 Gbps for decryption [182].

The new code was recompiled into the simulator and tested using that the previous control values from *drtestg.tcl* to produce identical results with zero delay.

Another snapshot of the code was taken at this point:

```
cp -p mac-802_11.h mac-802_11.h.dr2
cp -p mac-802_11.cc mac-802_11.cc.dr2
```

8.2.2 Further Enhancements

It was then decided to separately test the effects of the encryption delay and the decryption delay for various numbers of STAs independently. The code was further modified to allow the author to isolate the effects of encryption, decryption, or both, as described here. This additional code was again compiled into the simulator and utilised to produce the results herein.

The following code uses the new `CryptoTimer()` and `DstDecTimer()` classes that were added to the `MacTimer` files, *~ns2/mac/mac-timers.h* and *~ns2/mac/mac-timers.cc*, as already described in Chapter 7 [sections 7.3.3 and 7.3.4].

The `Mac802_11::transmit(Packet *p, double timeout)` routine was modified to replace the original:

```
        downtarget_->recv(p->copy(), this);
        mhSend_.start(timeout);
        mhIF_.start(txtime(p));
```

with:

```
/*
 *
 * Refined to enhance WLS and PWLS simulation.
 * The default MAC delay is 6.4us  but is not used by Mac/802_11.
 * We use MAC delay_ to hold the DECRYPTION delay for 128 bits.
 *
 * As a benchmark this uses the ratio of
 * 2.257 Gbps encryption to 1.722 Gbps decryption
 * So encryption = (1.722/2.257 =) 0.7629596810 * delay_
 *
 * Most MPDUs will have 3 addresses = 2+2+6+6+6+2+8 bytes of
 * MAC + LLC headers (+4 bytes FCS unencrypted) plus
 * 72 bits of preamble and 48 bits of PLCP header unencrypted
 * (but ns-2 always adds all 4 addresses to the size, but not
 *  the LLC header, so 120 bits + 2+2+6+6+6+6+2 + 4 bytes
 *  = 15 + 30 + 4 bytes = 49 bytes = 392 bits added to the MSDU)
 *
 * So frames have 120 bits unencrypted at 1 MHz (ns-2) = 120 us
 * (but really 72 bits at 1 MHz + 48 at 2 MHz = 72 + 24 = 96 us)
 * Plus a 50 us DIFS and 0-31 x 20 us slots = 146-766 us before
 * the first encrypted bit is needed (min 146 us avg 456 us)
 *
 * That means any encryption rate over 876,712 bps will always
 * have the first encryption block in time, which is less than
 * lowest data rate, so only data rate is really relevant.
 *
 * As long as encryption is faster than data rate no effects.
 * If encryption time is grater than tx_time + DIFS (0 slots)
 * (worst case) then worst case delay is encryption time less
 * (tx_time + DIFS)
 *
 * ALSO, the retransmission timers must account for receiver
 * decryption time (See recv(), below)
 */

// first adjust the timeout with RECEIVE delay at the other end
// This is duplicated (index_ reversed) from the recv() function
// See the recv() function for the details
// Receive the PLCP and the first 128 bits
double addrdelay = 0.00012237037037;

// Special - no decryption delay (no matter what was seeded)
// delay_ = 0;
// addrdelay = 0;
// Special - make decryption 6 Mbps slowing by 1 kbps per second
// delay_ = 128/(60000000-Scheduler::instance().clock()*1000000);

// Add decryption delays as appropriate (reverse logic to recv())
if(index_ == 0) {
        // This is the AP for my simulations
        // sending to a STA with one key
        addrdelay += delay_;
} else {
        // This is just a STA
        // Sending to the AP for my simulations
        struct hdr_cmn *hdr = HDR_CMN(p);
        if(hdr->size() == 29) {
                // This is an ACK or a CTS with no src
                // Try half the keys on average to decode this
                // Adjust this for the number of STA in the sim
                //      ((#MULTIkeys + #STA) / 2)
                addrdelay += (((1.0 + 1.0) / 2.0) * delay_);
        } else {
                struct hdr_mac802_11* mh = HDR_MAC802_11(p);
                u_int32_t dst = ETHER_ADDR(mh->dh_ra);
                u_int32_t src = ETHER_ADDR(mh->dh_ta);
                //      (#MULTIkeys + STAindex)
                addrdelay += ((1.0 + src) * delay_);
                // Special - simulate the XXXth node only
                // addrdelay += ((1.0 + 1.0) * delay_);
        }
}
// If decrypting takes longer than receiving, add the difference
if(addrdelay > txtime(p)) timeout = timeout + addrdelay - txtime(p);
```

```
// Now back to the business of this module - TRANSMIT delay
// Now work out any transmission delays...

// Special - no encryption delay (no matter what was seeded)
// delay_ = 0;
// Special - make encryption 6 Mbps slowing by 1 kbps per second
// delay_ = 128/(60000000-Scheduler::instance().clock()*1000000);

// Total time to encrypt the MPDU less FCS (less dst addr for PWLS)
// Set the MPDU size in the Tcl scripts to the next 128 bit boundary
// ns-2 adds 392 bits of PLCP, MAC and FCS
// Take off 120 PLCP and 32 FCS (152 bits) -> 240 encrypted bits
// or only 240 - 48 = 192 encrypted bits added for PWLS
// So 2048 bytes = 2068 (DSDV error) = 16544 bits
// PWLS 16544 + 192 = 16736 -> 131 x 128 - 196 = 16572 = 2072 bytes
//  WLS 16544 + 240 = 16784 -> 132 x 128 - 240 = 16656 = 2082 bytes
// Each less 20 bytes that will be added by the DSDV routing here
u_int32_t size = HDR_CMN(p)->size();
// Encrypting is only 1722/2257 as expensive as decrypting
// So encrypting is 0.7629596810 * delay_ per 128-bit block
//               or delay_ / 167.7677120  per bit
// PWLS Encryption time
//double crptdelay = ((double)size * 8.0 - 200) * delay_ / 167.7677120;
// WLS Encryption time
double crptdelay = ((double)size * 8.0 - 152) * delay_ / 167.7677120;
// If the encryption time is greater than the transmission time
// plus a DIFS up to the start of the last 128 bits (16 bytes)
double lastblock = phymib_.getDIFS() + txtime(p) - (128 / dataRate_);
if(crptdelay > lastblock) {
// then encrypt some before sending
        delayedPacket_ = p->copy();
        Handler *postdecrypt_ = new CryptoTimer(this);
        crptdelay = crptdelay - lastblock;
        timeout = timeout + crptdelay;
Scheduler::instance().schedule(postdecrypt_, delayedPacket_, crptdelay);
} else {
        downtarget_->recv(p->copy(), this);
        mhIF_.start(txtime(p));
        }
        mhSend_.start(timeout);
}
```

Where the `CryptoTimer` handler is:

```
Mac802_11::cryptoHandler()
{
        /*
         * Transmit packet after (starting) encryption
         */
        downtarget_->recv(delayedPacket_, this);
        mhIF_.start(txtime(delayedPacket_));
}
```

The `Mac802_11::recv(Packet *p, Handler *h)` routine was modified to replace the original:

```
                mhRecv_.start(txtime(p));
```

with:

```
                /*
                 * mhRecv_.start(txtime(p));
                 *
                 * Refined to enhance WLS and PWLS simulation.
                 * The default MAC delay is 6.4us but is not used by Mac/802_11.
                 * We use MAC delay_ to hold the DECRYPTION delay for 128 bits.
                 *
                 * Decrypting the whole MSDU for PWLS happens in
                 * recvDATA, before handing on to uptarget_
                 *
                 * But before we can do address filtering in WLS
                 * we must first recieve and decrypt the addresses
                 * which will be in the first cipherblock
                 *
                 * Most MPDUs will have 3 addresses = 2+2+6+6+6+2+8 bytes of
                 * MAC + LLC headers (+4 bytes FCS unencrypted) plus
                 * 72 bits of preamble and 48 bits of PLCP header unencrypted
                 * (but ns-2 always adds all 4 addresses to the size, but not
                 *  the LLC header, so 120 bits + 2+2+6+6+6+6+2 + 4 bytes
```

```
*  = 15 + 30 + 4 bytes = 49 bytes = 392 bits added to the MSDU)
*
* So frames have 120 bits unencrypted at 1 MHz (ns-2) = 120 us
* (but really 72 bits at 1 MHz + 48 at 2 MHz = 72 + 24 = 96 us)
* before receiving the first cipherblock,
* taking only 128 / 54 Mbps = 2.370370370 us,
* which must then be decrypted, taking a delay_ for
* every key attempted, BEFORE any filtering decisions.
*
* AFTER filtering, any frame for this STA can be ACKed
* after FCS (receive complete)
*
* Excessively slow decryption rates may delay MSDU's being
* passed to higher layers but will not affect the WM
* throughput except for short packets that take longer
* to decrypt the addresses than to finish receiving
* the frame
*
* Receiving the first cipherblock needs preamble at 1 Mbps,
* PLCP at PLCP rate, then the first 128 bits at the data rate
* 72bits/1Mbps + 48bits/2Mbps + 128/54Mbps
* but ns uses a single PLCPDataRate
* (72bits + 48bits = 120bits)/1Mbps + 128/54Mbps
* = 0.00012s + 0.000002370370370s = 0.0001223703704s
* Decrypting the first block will take a further delay_
*
* Decrypting the address will require multiple itterations
* of decrpting the first 10 octets for each of the known "keys"
* which will require the first (block) 128 bits for each key
* 128 / 1722000000 = 0.00000007433217 seconds
* and will require min 1, max all keys tested
* receiving from STA1 = 0.00000007433217 seconds
* receiving from STA2 = 0.00000014866434 seconds
* receiving from STA5 = 0.00000037166086 seconds
* receiving from STA10 = 0.00000074332172 seconds
* receiving from STA25 = 0.00000185830430 seconds
* receiving from STA50 = 0.00000371660859 seconds
* receiving from STA100 = 0.00000743321718 seconds
```

```
 * receiving from STA150 = 0.00001114982578 seconds
 * receiving from STA250 = 0.00001858304297 seconds
 *
 * For an ACK frame, that is frame control 16bits,
 * duration 16bits, address 48bits and fcs 32bits = 112bits,
 * this is longer than the transmission time
 * and must be added here.
 * ACKs & CTS are 29bytes (120bits PLCP + 112bits MPDU)
 * All data frames in these simulations are 2117+ bytes
 * 15B PLCP, 2 FC, 2 dur, 4x6 addrs, 2 SC, MSDU, 4 FCS
 * where MSDU = 20 hdr + 2048+ data
 */

// Receive the PLCP and the first 128 bits
double addrdelay = 0.00012237037037;

// Special - no decryption delay (no matter what was seeded)
// delay_ = 0;
// addrdelay = 0;
// Special - make decryption 6 Mbps slowing by 1 kbps per second
// delay_ = 128/(60000000-Scheduler::instance().clock()*1000000);

// Add decryption delays as appropriate
if(index_ == 0) {
        // This is the AP for my simulations
        if(hdr->size() == 29) {
                // This is an ACK or a CTS with no src
                // Try half the keys on average to decode this
                // Adjust this for the number of STA in the sim
                //     ((#MULTIkeys + #STA) / 2)
                addrdelay += (((1.0 + 1.0) / 2.0) * delay_);
        } else {
                struct hdr_mac802_11* mh = HDR_MAC802_11(p);
                u_int32_t dst = ETHER_ADDR(mh->dh_ra);
                u_int32_t src = ETHER_ADDR(mh->dh_ta);
                //     (#MULTIkeys + STAindex)
                addrdelay += ((1.0 + src) * delay_);
                // Special - simulate the XXXth node only
```

```
                        // addrdelay += ((1.0 + 1.0) * delay_);
                }
        } else {
                // This is just a STA with one key
                addrdelay += delay_;
        }
        // Wait for receiving or decrypting, whichever is longer
        if(txtime(p) > addrdelay) addrdelay = txtime(p);
        mhRecv_.start(addrdelay);
```

After the Receive Timer (the time to receive the frame from start to finish) expires without collision, then the `Mac802_11::recvDATA(Packet *p)` routine was modified to replace the original, in three places, firstly decrypting the rest of the frame (only at least the first block is decrypted during the receive time so that the address filtering can happen in this routine), by inserting after:

```
/*
 *  Pass the packet up to the link-layer.
 *  XXX - we could schedule an event to account
 *  for this processing delay.
 */
```

with:

```
/*
 *  Pass the packet up to the link-layer.
 *  XXX - we could schedule an event to account
 *  for this processing delay.
 */
/*
 * Decrypting the whole frame is completed here
 * if not already, before handing on to uptarget_
 * All the PLCP and decrypting the first block
 * have been done already in recv().
 * If we have got here, then this packet is for us
 * and we already have the right key.
 * The total decryption time is MSDU * delay_
 * But we have been decrypting ever since we got the
 * first 16 bytes, which have at least been decrypted.
 * But there were another 14 bytes of header already stripped.
```

```
         */

        // Special - no decryption delay
        // delay_ = 0;
        // Special - make decryption 6 Mbps slowing by 1 kbps per second
        // delay_ = 128/(60000000-Scheduler::instance().clock()*1000000);

        double addrdelay = 0.0;
        // Time to decrypt (size+14)/16 128-bit blocks.
        addrdelay = ((double)size + 14.0) / 16.0 * delay_;
        // Less time already spent while receiving the
        // rest of the packet after the first 16 bytes.
        addrdelay -= ((double)(size + 14) * dataRate_);
        // Either way, there is at least the last block to decrypt
        if(addrdelay < delay_) addrdelay = delay_;
```

and then for broadcast frames, replace the original:

```
                if (dst == MAC_BROADCAST) {
                        uptarget_->recv(p->copy(), (Handler*) 0);
                }
```

with:

```
                if (dst == MAC_BROADCAST) {
                        /*
                         * uptarget_->recv(p->copy(), (Handler*) 0);
                         */
                        if (addrdelay) {
                                delayedPacket_ = p->copy();
                                Handler *postdecrypt_ = new DstDecTimer(this);
                                Scheduler::instance().schedule(postdecrypt_, delayed
                        } else uptarget_->recv(p->copy(), (Handler*) 0);
                }
```

and finally for unicast frames, replace the original:

```

        uptarget_->recv(p, (Handler*) 0);

}
```

with:

```
        /*
         * uptarget_->recv(p, (Handler*) 0);
         */
        if (addrdelay) {
                delayedPacket_ = p->copy();
                Handler *postdecrypt_ = new DstDecTimer(this);
                Scheduler::instance().schedule(postdecrypt_, delayedPacket_, addrdelay);
        } else uptarget_->recv(p->copy(), (Handler*) 0);
}
```

Where the `DstDecTimer` handler is:

```
Mac802_11::dstDecHandler()
{
        /*
         * Pass up packet after completing decyption
         */
        uptarget_->recv(delayedPacket_, (Handler*) 0);
}
```

8.2.3 Snapshots of the Code

These modifications were saved (with the zero delay ‘specials’ turned on) as:

```
cp -p mac-802_11.h mac-802_11.h.specials
cp -p mac-802_11.cc mac-802_11.cc.specials
```

but are otherwise identical to the (no ‘specials’ active) final code in:

```
mac-802_11.h.final
mac-802_11.cc.final
mac-timers.h.final
mac-timers.cc.final
```

available from the author, the Queensland University of Technology or online at http://www.antacs.com

8.2.4 Compiling and Validation

The new code was recompiled into the simulator, with:

```
cd /usr/src/ns-2.33
make clean
make
make install
```

and validated with:

```
./validate >>validate.out.final 2>&1
```

resulting in:

```
(Validation can take 1-30 hours to run.)
Sun Apr 19 16:48:18 EST 2009
...
Test output agrees with reference output
All test output agrees with reference output.
Sun Apr 19 18:11:22 EST 2009
These messages are NOT errors and can be ignored:
    warning: using backward compatibility mode
    This test is not implemented in backward compatibility mode

validate overall report: all tests passed
```

8.3 Verifying the Simulator

The simulator was then tested using that the previous control scenarios from *drtestg.tcl*, *drtestl.tcl*, *drtestn.tcl* and *drtestp.tcl* were configured in *drtestwa.tcl* through *drtestwd.tcl*, as shown here, to produce identical results with zero delay, as shown in Figures 8.1 through 8.4.

```
# w series uses the new code, no multiplier, padded lengths, fixed timeouts
# wa validation for zero delay against control data from drtestg
#
# These simulations are using the MAC delay_ to hold the DECRYPTION delay
# The default MAC delay is 6.4us  but is not used by Mac/802_11
# The default HERE is to use decryption at 1.722 Gbps for each 128-bit block
# 128 / 1722000000 = 0.00000007433217190s = 74.33217190ns
# ENCRYPTION is typically faster and I use 2.257 Gbps for each 128-bit block
# The modified Mac/802_11 uses (1722/2257=)0.7629596810*delay_ for encryption
# Mac/802_11 set delay_ 74.33217190ns
#
# But this case is the zero delay validation
# Simulator set to zero irrespective of what is here
Mac/802_11 set delay_ 0us
```

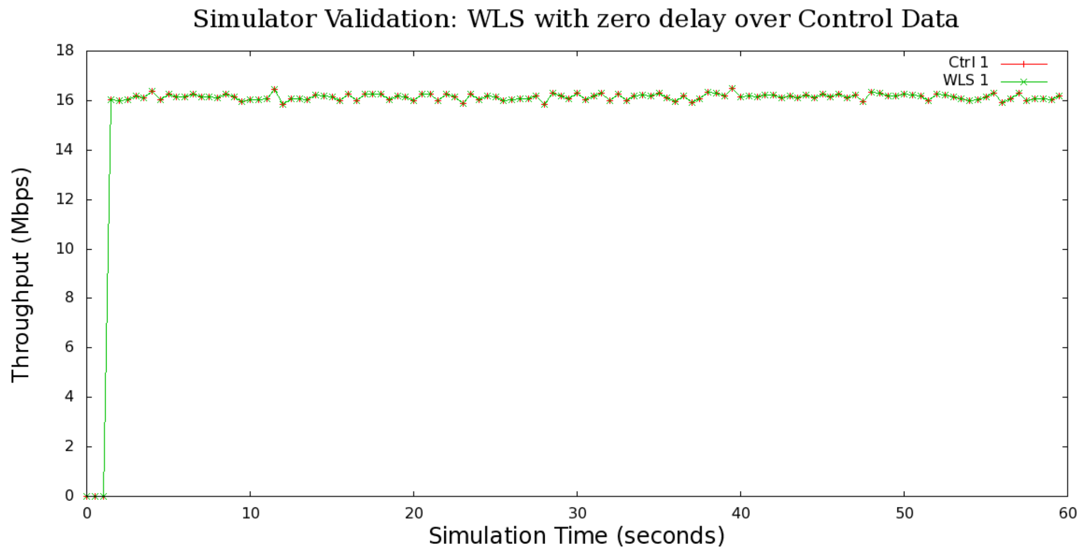


Figure 8.1: Enhanced WLS Simulator using a 16.16 Mbps CBR Source

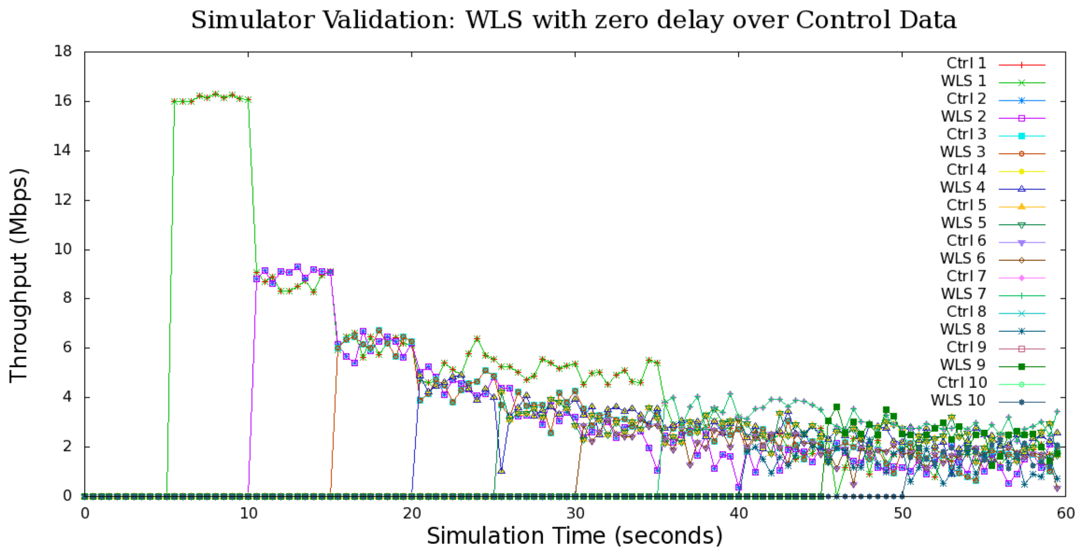


Figure 8.2: Enhanced WLS Simulator with ten 16.16 Mbps CBR nodes

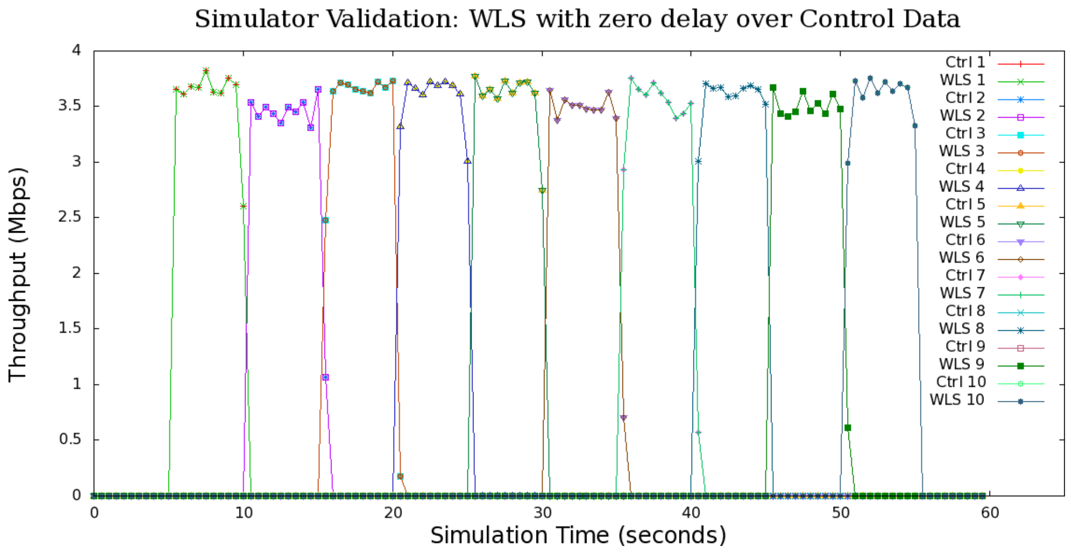


Figure 8.3: WLS with ten TCP (2.0 MB FTP) sessions

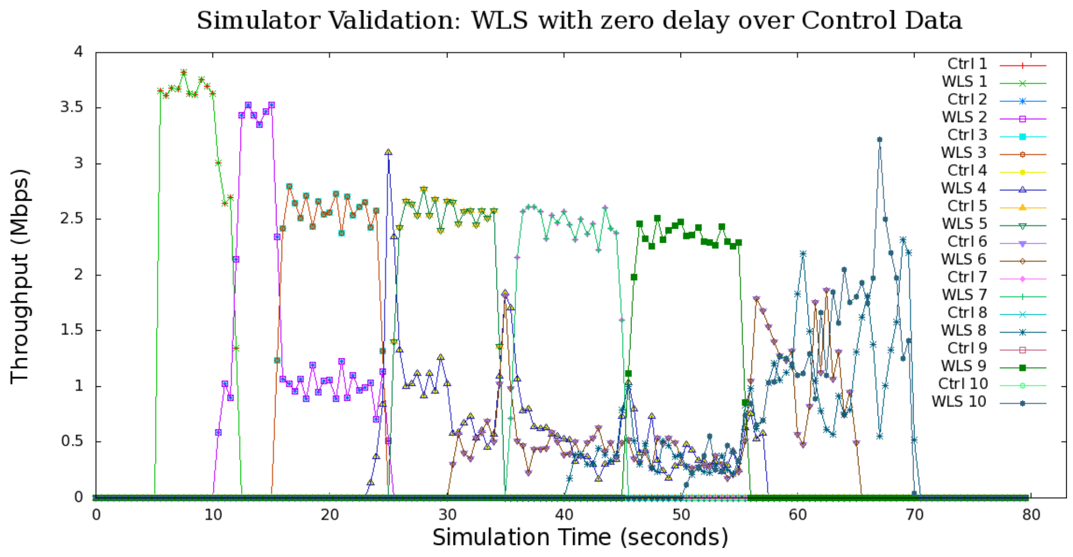


Figure 8.4: WLS with ten overlapping TCP (2.6 MB FTP) sessions

8.4 Enhanced Results

Having validated the simulator results for normal WLAN operation, the Ten CBR Node scenario from *drtestl.tcl* was configured for the new simulator in *drtestwe.tcl*, with:

```
Mac/802_11 set delay_ 74.33217190ns
```

The delay shown here, for decrypting a 128-bit block under AES CCMP, provides an encryption rate of 2.257 Gbps and a decryption rate of 1.722 Gbps. The results of using ‘full’ WLS with this rate of encryption/decryption are shown in Figure 8.5.

Figure 8.5 indicates that with an encryption/decryption rate of 2.257/1.722 Gbps, invoking WLS *has no effect* on network throughput at all, producing an exact replica of the control data.

To aid the visualisation of these results, this scenario was continued using the “Average Throughput per Node”, of *drtestm.tcl*. This was configured for the same encryption/decryption rate in *drtestwf.tcl*, with the result shown in Figure 8.6.

The decryption delay was then increased to match the 2 µs maximum propagation delay that had destroyed the network throughput in the previous tests. This was configured in *drtestwg.tcl*, with:

```
# Mac/802_11 set delay_ 74.33217190ns
```

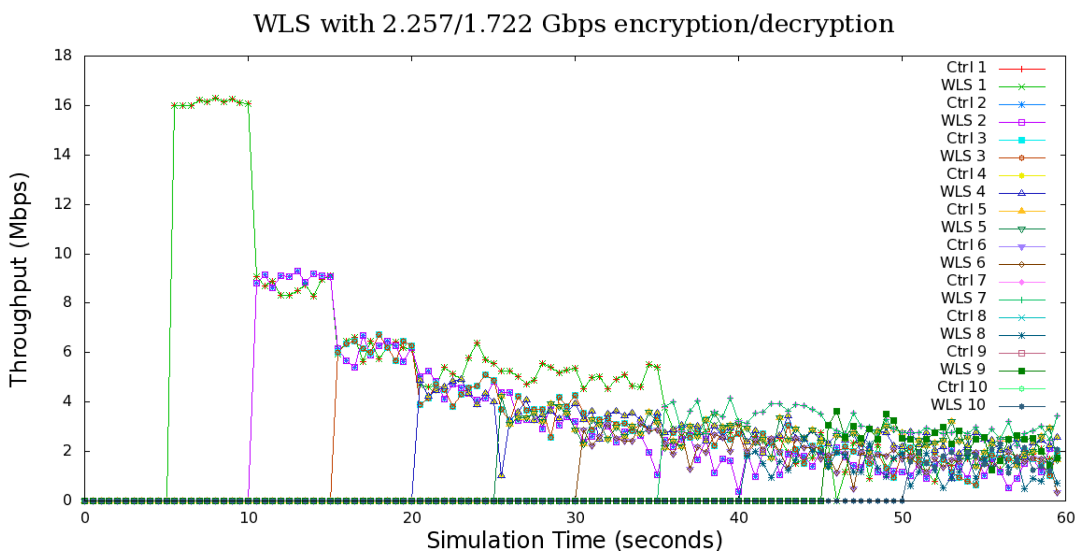


Figure 8.5: WLS with ten nodes — 2.257/1.722 Gbps encryption/decryption

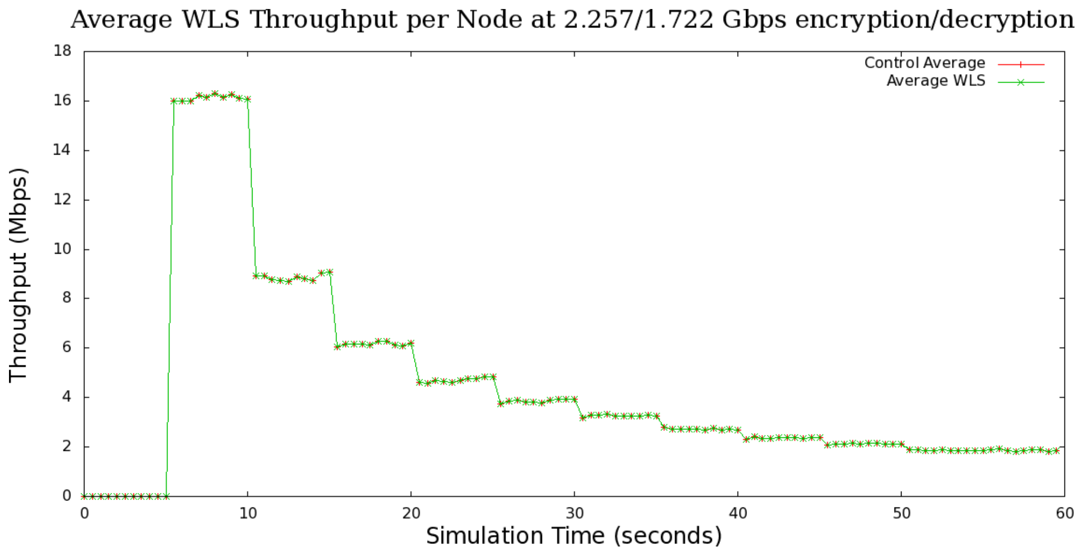


Figure 8.6: Average WLS Throughput per Node — 2.257/1.722 Gbps crypto

```
Mac/802_11 set delay_ 2.0us
```

This the decryption delay and represents a mere 64 Mbps decryption rate or an 84 Mbps encryption rate (1.53 µs encryption delay). The results of using WLS with this rate of encryption/decryption are shown in Figure 8.7.

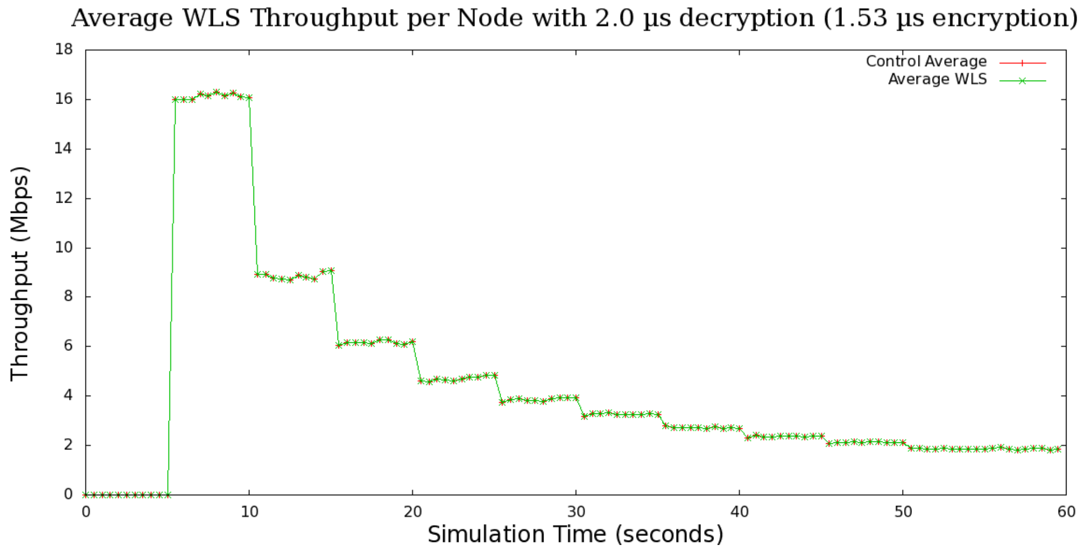


Figure 8.7: Average WLS Throughput — crypto slowed to 84/64 Mbps

Again, Figure 8.7 indicates that even with an encryption/decryption rate of only 84/64 Mbps, invoking WLS still as no effect on network throughput at all.

The decryption delay was then increased further, so that the encryption delay would to match the 2 µs maximum propagation delay. This was configured in *drtestwh.tcl*, with:

```
# Mac/802_11 set delay_ 74.33217190ns
# Mac/802_11 set delay_ 2.0us
Mac/802_11 set delay_ 2.6213705us
```

This now represents a 64 Mbps encryption rate (2 µs for 128 bits) or only a 49 Mbps decryption rate. The results of using WLS with this rate of encryption/decryption are shown in Figure 8.8.

Once again, Figure 8.8 indicates that even with an encryption/decryption rate of only 64/49 Mbps, invoking WLS still as no effect on network throughput at all.

At 3.0 µs, the result did not perfectly match the control, but the difference was negligible. By 5.0 µs, the effects became significant. This was configured in *drtestwi.tcl*, with:

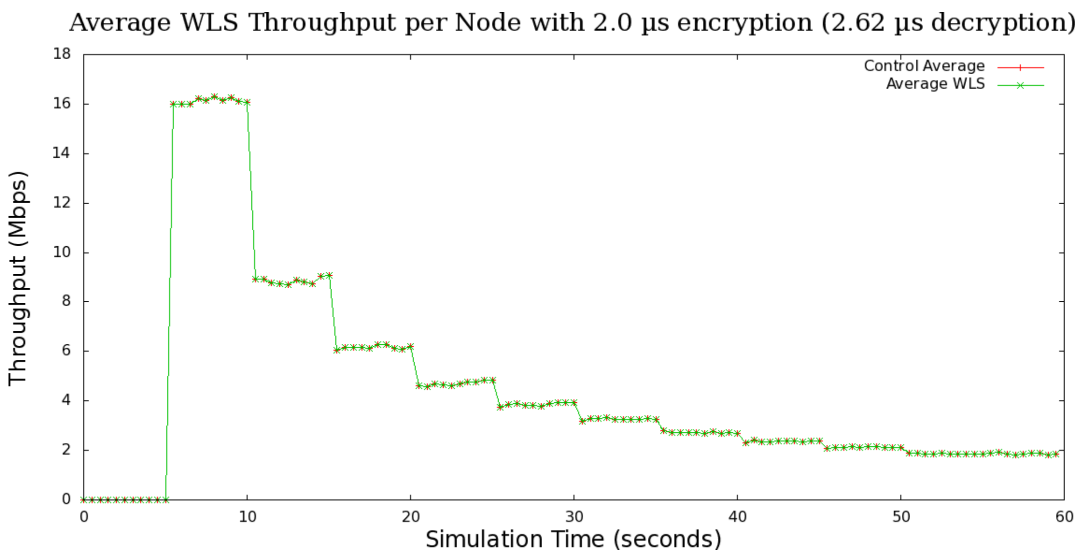


Figure 8.8: Average WLS Throughput — crypto slowed to 64/49 Mbps

```
# Mac/802_11 set delay_ 74.33217190ns
# Mac/802_11 set delay_ 2.0us
# Mac/802_11 set delay_ 2.6213705us
Mac/802_11 set delay_ 5.0us
```

This now represents only a 25.6 Mbps decryption rate or a 33.6 Mbps encryption rate. The results of using WLS with this rate of encryption/decryption are shown in Figure 8.9.

Figure 8.9 shows a noticeable deviation from the control data for decryption rates below 40 Mbps.

At 15.0 µs, the throughput achieves less than half the control value. This was configured in *drtestwj.tcl*, with:

```
# Mac/802_11 set delay_ 74.33217190ns
# Mac/802_11 set delay_ 2.0us
# Mac/802_11 set delay_ 2.6213705us
# Mac/802_11 set delay_ 5.0us
Mac/802_11 set delay_ 15.0us
```

This now represents a decryption rate significantly less than 10 Mbps. The results of using WLS with these rates are shown in Figure 8.10.

Figure 8.10 shows a major deviation from the control data for decryption rates below 10 Mbps.

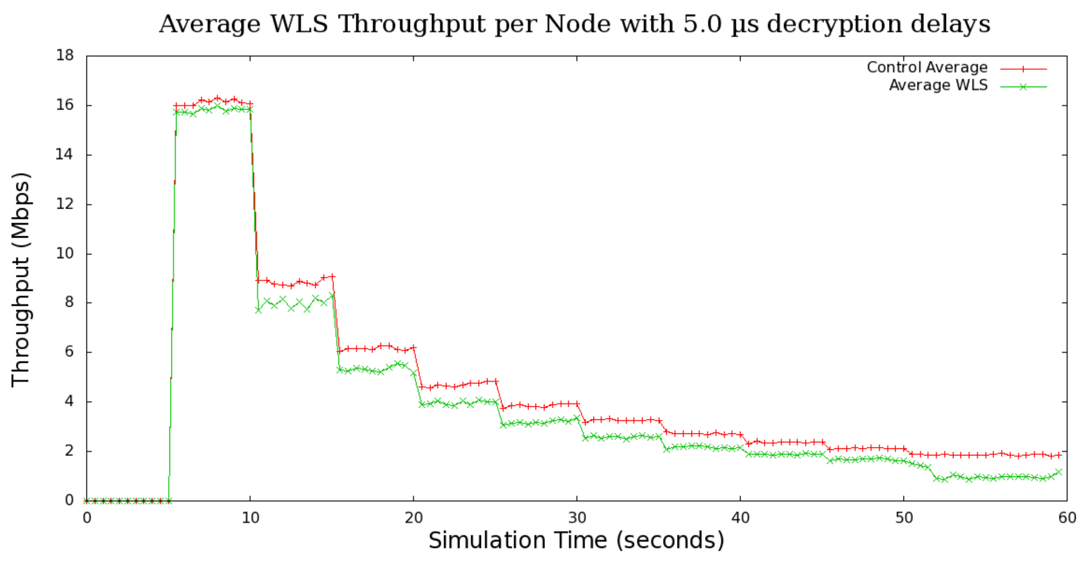


Figure 8.9: Average WLS Throughput — crypto slowed to 33.6/25.6 Mbps

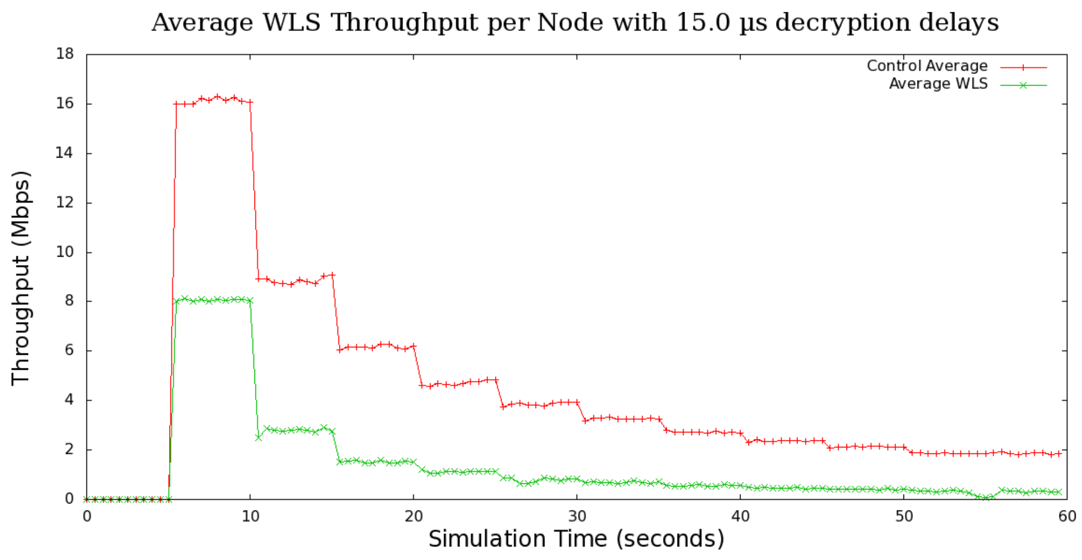


Figure 8.10: Average WLS Throughput — crypto slowed to 11.18/8.5$\dot{3}$ Mbps

Finally, at 50.0 µs, the throughput achieves only a small fraction of the control value. This was configured in *drtestwk.tcl*, with:

```
# Mac/802_11 set delay_ 74.33217190ns
# Mac/802_11 set delay_ 2.0us
# Mac/802_11 set delay_ 2.6213705us
# Mac/802_11 set delay_ 5.0us
# Mac/802_11 set delay_ 15.0us
Mac/802_11 set delay_ 50.0us
```

This now represents a decryption rate significantly of only 2.56 Mbps. The results of using WLS with these rates are shown in Figure 8.11.

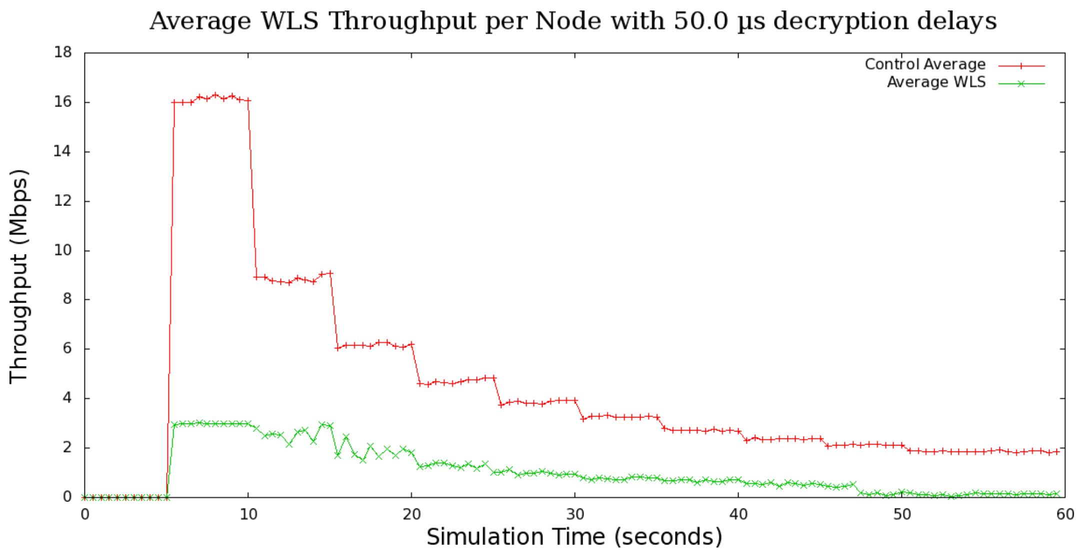


Figure 8.11: Average WLS Throughput — crypto slowed to 3.36/2.56 Mbps

Figure 8.11 shows the throughput limited to the decryption rate.

8.5 Detailed Rate Results

The simulator was then modified, using the code switches shown above, to vary the encryption and/or decryption rates during the simulation run. This was achieved using:

```
        // Special - make encryption 6 Mbps slowing by 1 kbps per second
        // delay_ = 128/(60000000-Scheduler::instance().clock()*1000000);
```

and:

```
        // Special - make decryption 6 Mbps slowing by 1 kbps per second
        // delay_ = 128/(60000000-Scheduler::instance().clock()*1000000);
```

The simulator was first modified to use no decryption delay and no encryption delay and was recompiled. *drtestwl.tcl* was configured to run the *drtestg.tcl* control scenario, but starting from time 0.0 and recording time as reverse encryption rate, as shown below.

```
# Mac/802_11 set delay_ 74.33217190ns
# The simulator will vary the encryption/decryption delay from 0 to 60 kbps.
...
    set f_($i) [open drtest$t$u-none1.tr w]
...
    exec xgraph drtestg-node1.tr drtest$t$u-none1.tr -geometry 800x400 &
...
    # $ns_ at $i "$cbr_($i) start"
    $ns_ at 0.0 "$cbr_($i) start"
...
        puts $f_($i) "[expr 60.0-$now] [expr $by_($i)/$time*8]"
...
$ns_ at 0.5 "record"
```

The results are shown in Figure 8.12.

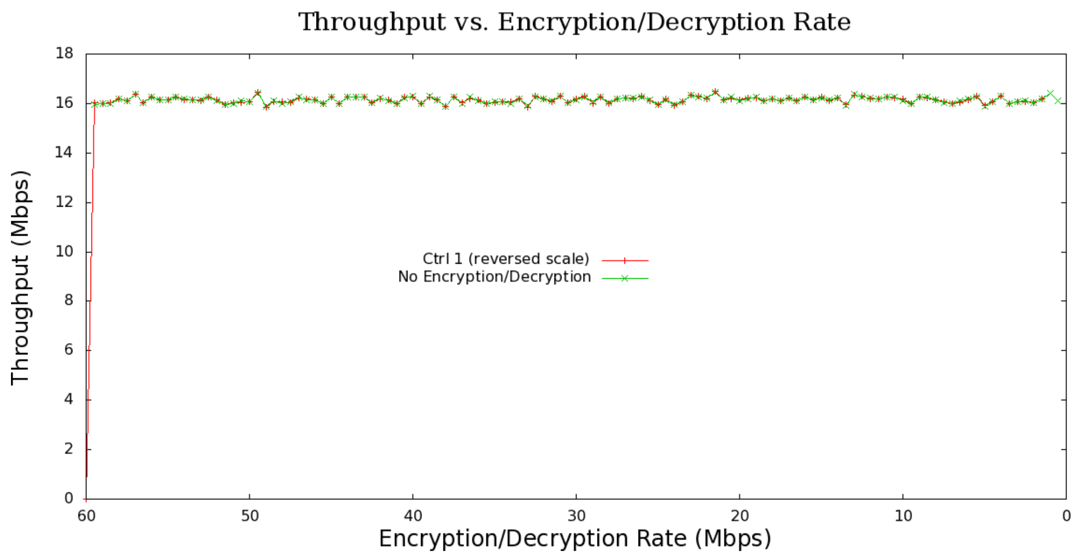


Figure 8.12: WLS Throughput with all delays disabled cf. Control Data

Figure 8.12 shows the results for no encryption or decryption are a mirror image (reversed scale) of the control data.

Next, the simulator was modified to use no decryption delay and the variable encryption delay and was recompiled. *drtestwm.tcl* was duplicated to run the same scenario as *drtestwl.tcl*, but this time recording into the file *drtestwm-encr.tr*.

The results are shown in Figure 8.13.

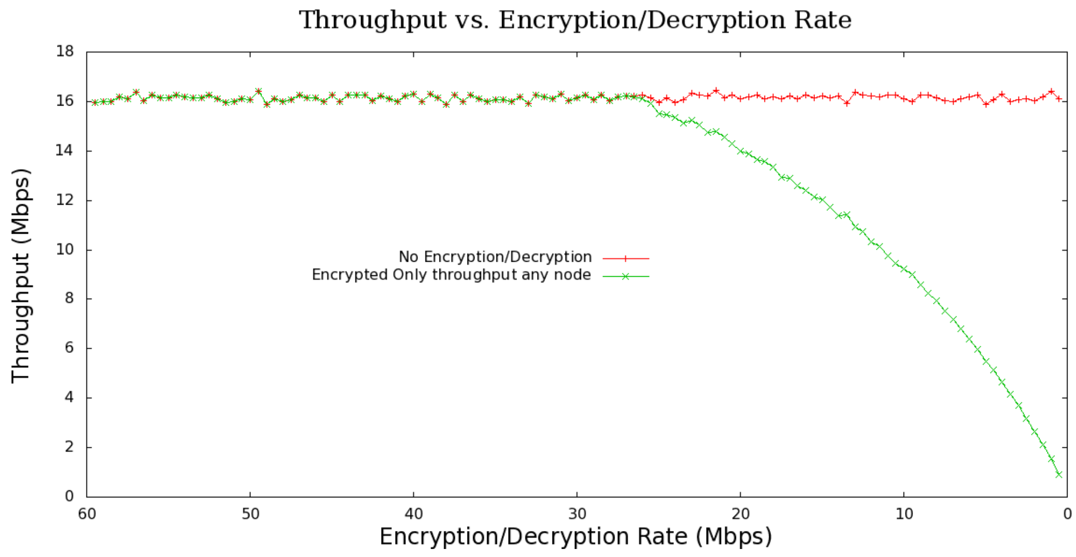


Figure 8.13: WLS Throughput vs. Encryption Rate (decryption delays disabled)

Next, the simulator was modified to use the variable decryption delay for the first node in the “key list”, but with no encryption delay and was recompiled. *drtestwn.tcl* was again duplicated to run the same scenario as *drtestwm.tcl*, but this time recording into the file *drtestwn-decr1.tr*.

The results are shown in Figure 8.14.

Next, the simulator was modified to use the variable decryption delay for the *tenth* node in the “key list”, but still with no encryption delay. *drtestwn.tcl* was modified to record into the file *drtestwn-decr10.tr*.

The results are shown in Figure 8.15.

This was repeated for the 100th node in the “key list” and *drtestwn.tcl* was modified to record into the file *drtestwn-decr100.tr*. The results are shown in Figure 8.16.

Finally, the simulator was modified to use the variable decryption delay for the 1000th node in the “key list” and *drtestwn.tcl* was modified to record into

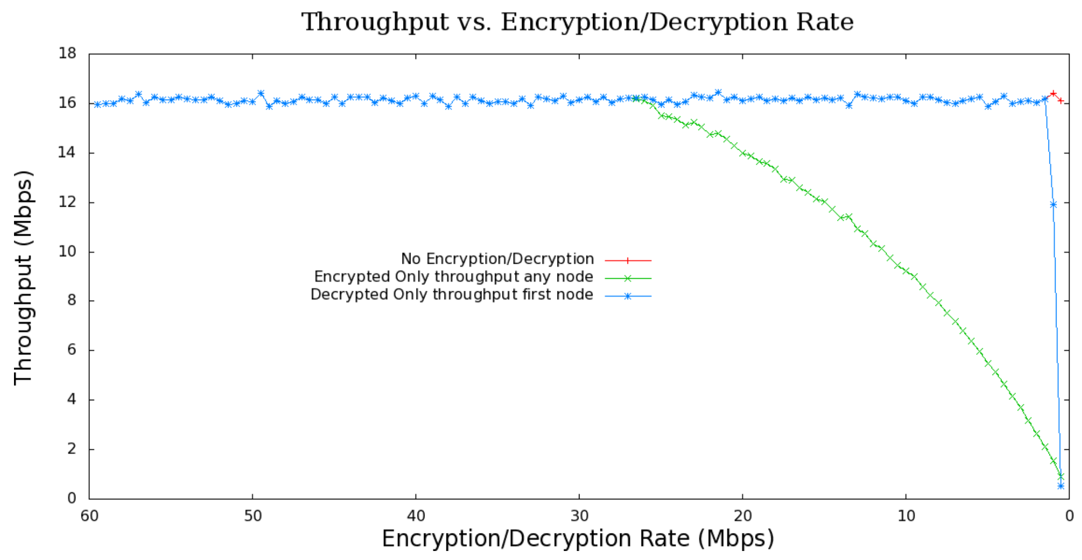


Figure 8.14: WLS Throughput vs. Decryption Rate for the First Unicast Key

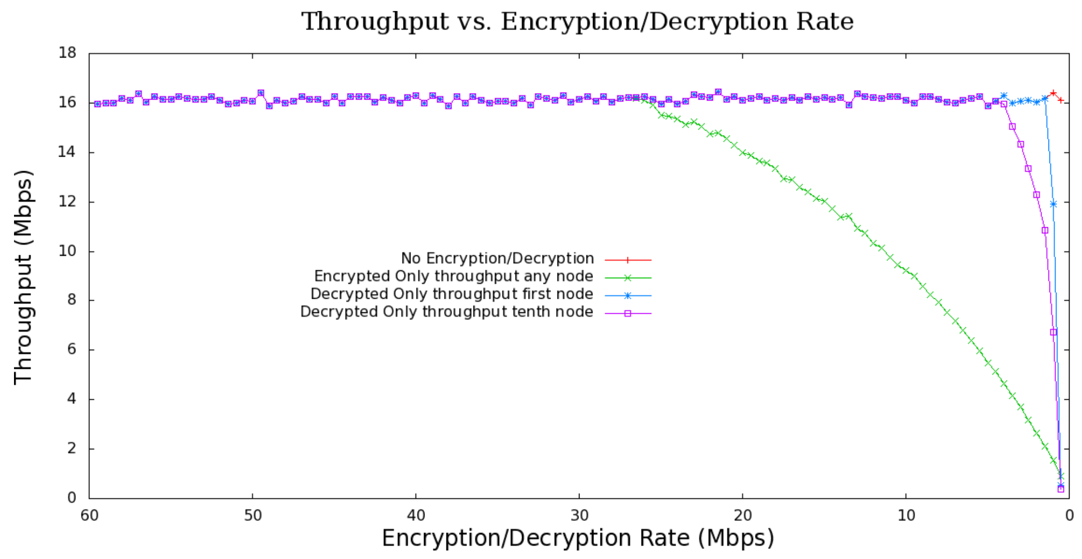


Figure 8.15: WLS Throughput vs. Decryption Rate for the Tenth Unicast Key

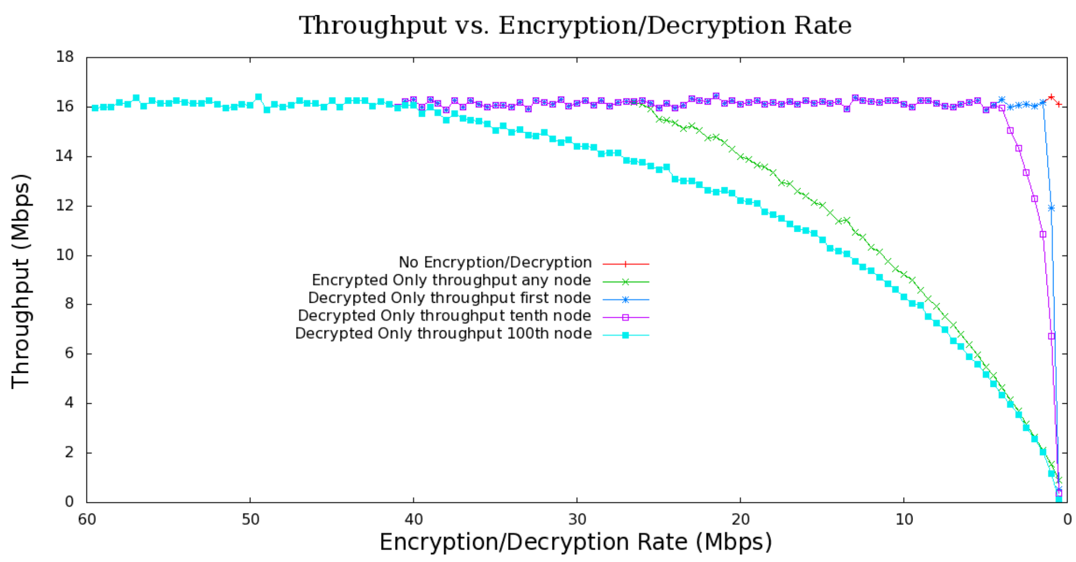


Figure 8.16: WLS Throughput vs. Decryption Rate for the 100th Unicast Key

the file *drtestwn-decr1000.tr*. The results are shown in Figure 8.17.

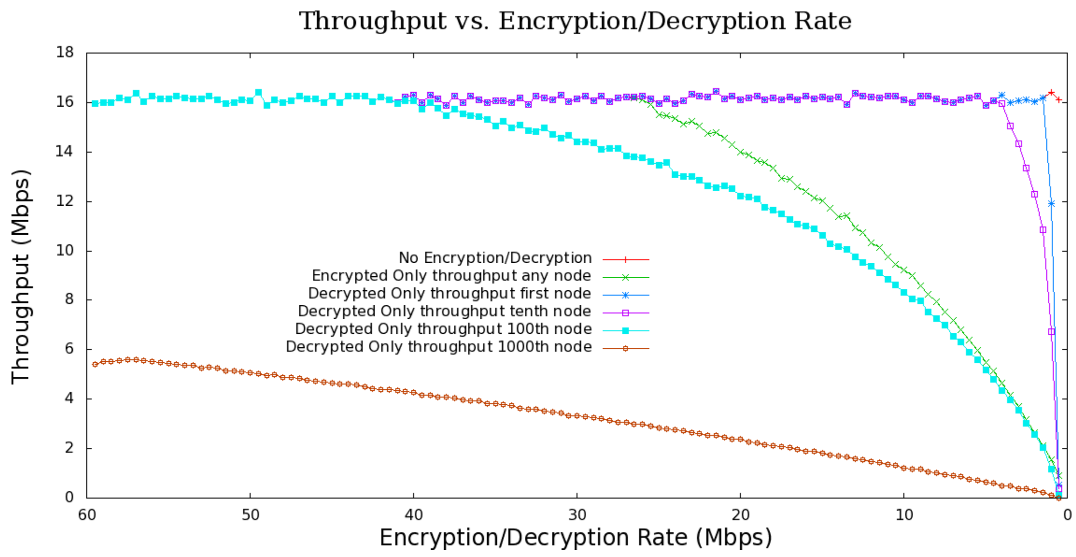


Figure 8.17: WLS Throughput vs. Decryption Rate for the 1000th Unicast Key

8.6 Discussion

As can be seen from these results, in our ten-node saturated-channel scenarios, the penalty for encrypting a packet at the MAC sub layer, as it is queued for transmission, has no effect on network throughput until the decryption rate drops below 40 Mbps.

In a two-node network, with one node transmitting to the other, any penalty in decrypting a received packet has no effect on network throughput until the decryption rate drops below approximately 2 Mbps. However, unlike encryption, which has the same penalty for every STA irrespective of the number of STAs, decryption must be performed using every available key.

In partial WLS, the decryption key is identified by the source address from the source address and so invokes the same penalty irrespective of the number of STAs in the network . However the complete WLS proposal does not provide an unencrypted source address and so every known a key must be checked in order to decrypt the packet, including the broadcast keys. As such, on average, approximately half of all of the keys must be tried for even an acknowledgement packet.

The simulations performed here used the same design for both ad hoc and infrastructure performance, simulating infrastructure networks by having the first `node_(0)` behave as a central AP receiving and transmitting to each of the other STAs.

In the system we are simulating here, an AP maintains an ordered a set of keys for each pairwise connection with its associated STAs in the same order that they associated. As such, the oldest associations will have their keys earlier in the AP's list and so suffer reduced decryption penalty at the AP. Conversely, the newest associations will result in keys at the end of the AP's list and thus suffer the greatest decryption penalty at the AP.

While this system is not fair for the last STAs in large networks and, in all practicality, realised implementations of this system would involve at least a randomising function — or more likely, a last-in first-out approach in attempting decryption keys. Such approaches, while making this fairer for individual STAs, will not affect the average overall throughput for a given number of STAs, and so do not affect the results we present here.

In a ten-STA network, the decryption penalty for traffic from the tenth node, the last to associate, and so the last key to be tried in the simulations, has no effect on network throughput until the decryption rate drops below approximately 4 Mbps, at which time network throughput dramatically falls, quickly are reaching zero for decryption rates falling below 1 Mbps.

Chapter 9

Conclusions

"All progress is precarious, and the solution of one problem brings us face to face with another problem" — Martin Luther King Jr., *Strength to Love*, 1963.

Finally This chapter concludes the work, summarizing the results of this work, as well as defining areas of future endeavour in this field.

9.1 Summary

This research has investigated the current state of the art in IEEE 802.11 wireless network security, including current weaknesses identified in Pre-Shared Keys, TKIP, the use of TSNs and the ability to select weak configurations based on the vendor offerings that permit insecure combinations, compromising what would otherwise be a Robust Security Network.

This work has looked at the security of the low level protocols and, in particular, vulnerabilities in and possible attacks on Control and Management frames in IEEE 802.11 WLANs. Two principal options for protecting the entire data link layer, including all of the MAC sub-layer headers, where proposed and investigated, along with a number of variants.

While acknowledging that the key establishment requirements may reduce the general utility of these proposals, it is believed that this limitation does not preclude deployments in a great many situations where this additional level of security is desired, or indeed required.

The major concern therefore was the impact of this additional cryptographic load at such a low level in the protocol stack.

A series of simulations were conducted to establish a baseline commensurate with real-world observations of WLAN performance. From this baseline, a detailed set of control data was generated providing communications links with Constant Bit-Rate (CBR) sources: in an uncontended medium, contending with another CBR STA, in scenarios of close and distant ranges, contending with many high-rate CBR sources, as well as simulated TCP data, both to and from an AP and including multiple overlapping TCP sessions.

These controls were then used to compare with and analyse the results of simulations of the new proposed protocols.

When initial simulations yielded results effectively identical to the controls, the simulations were adjusted to better recreate the physical realities of the proposals. An extensive set of simulations of these new protocols where conducted and analysed, confirming that the penalties invoked by applying cryptographic protections at this low level of the protocol stack did not affect network throughput for reasonable rates of encryption.

The data did however show, that exceptionally slow encryption rates and, in particular, low decryption rates at the receiver, many induce sufficient delay in the transmission of, or the response to, packets — resulting in retransmissions and hence retransmission failures, at which point, throughput is decimated, as all unicast traffic effectively fails, with only broadcast traffic being successful.

In light of these results, the simulated protocols were amended to account for delays both in transmission and particularly decryption by the receiver, in order to respond to unicast transmissions destined for that node, to permit a reply within the new time frame. These additional enhancements were then tested and the analysis of the results further supported the viability of these proposals.

While not necessarily appropriate in open networks permitting access by unknown STA without prior credentials, for those networks where the only permitted STA have already been provided with the appropriate credentials, then the Wireless Link Security proposed here may well be a viable solution, with the appropriate development.

9.2 Further Work

This research has shown that encrypting the entire data link layer can be a viable proposition. However, there are still a great many issues to be considered before these proposals could lead to a workable solution.

These proposals are based on the assumption that all of the stations already hold the required Public Key Infrastructure (PKI) credentials necessary to establish the pairwise unicast and groupwise broadcast keys necessary to initiate these protocols. However, no consideration has yet been made as to how these credentials are distributed, nor how such credentials would be used to establish such keys. This is a matter for future research in this area.

This research has only dealt with the established wireless network protocols, such as IEEE 802.11b and/or IEEE 802.11g, but has not investigated the effects of recent amendments in such areas as: adding quality of service, fast handoff for roaming or the emerging IEEE 802.11n High Throughput amendments [77] and the effects of MSDU-aggregation or block acknowledgements.

Moreover, this work has only touched on the practical vagaries of the implementing such protocols in hardware. Even with the extensive knowledge base and open code of groups such as the Linux wireless development community and the flexibility of chip sets, such as the Atheros wireless hardware, the intricacies of encrypting the entire MPDU, including headers, are likely to induce their own range of unique difficulties.

Along with these major areas, further research in these and any other wireless networking protocols, will require enhancements to the base simulation tools to account for the current advancements in wireless networking itself. While the *ns-2* simulator still remains more popular than the *ns-3* successor, the majority of the development effort in the simulator itself now appears to be squarely focused on advancing *ns-3* as the replacement for *ns-2* [160].

In addition, further work is required to verify this work against fully functioning infrastructure mode simulations, as well as without and the faulty *ns-2* DSDV routing protocol that added an extra twenty bytes to every packet passed in these simulations.

9.3 Final Remarks

While the spectre of encrypted MAC addresses traversing the wireless medium, seemingly unable to find a home, at first appears a major challenge, this work has shown that this is not unreasonable and, in the right situation, may well be a desirable feature and in the right application, entirely appropriate, in view of the applicable risk profile that such protocols maybe addressing .

As in all matters of information security, the application of any security control should always be the result of a risk-based decision applicable to the particular situation being addressed. The application of any security control, simply in the name of security, may well and add unnecessary burden for minimal gain, and if not done in response to or in accordance with, an appropriately designed security architecture, is unlikely two best support the purposes of the infrastructure in meeting the business needs.

Appendix A

Developing the Tool Platforms to Test the Thesis

This author's choice of the *Red Hat / CentOS / Fedora* OS family is described in subsection 6.1.1 of Chapter 6. A reader wishing to replicate these experiments may well prefer a *Debian* distribution or a derivative, such as *Ubuntu*, or some other Unix-like OS, since command-line and basic GUI operations are the same on any distribution.

The following describes the set-up on a Fedora distribution and may not involve identical commands and configuration on other systems.

A.1 Preparation, Installation and Configuration of the Tools

On the 29th March 2008, the initial preparations were commenced for the development of the simulation tools. Once a stable platform has been prepared, originally *Fedora 7* in this case, the set up of *ns-2* proceeded as follows. Version *ns-2.32* was initially chosen, but was immediately replaced by *ns-2.33*, being the most current stable release, as at 31st March 2008. The "*ns-allinone-2.29.3*" version was not used as the *ns-2.3x* versions contained required wireless additions for this work.

A.1.1 Preparing the Environment

The first step was to check that the development environment was complete and install any missing tools.

```
rpm -q wget
rpm -q gcc-c++
rpm -q libX11-devel
rpm -q xorg-x11-proto-devel
rpm -q libXt-devel
rpm -q libXmu-devel
```

A.1.2 Download and Install Tcl/Tk

Next *Tcl* and *Tk* and Object-oriented-Tcl, *OTcl* were prepared. The latest releases of *Tcl* and *Tk* were 8.5.2 and the latest *OTcl* was 1.13. These were obtained, compiled and installed as follows.

```
cd /usr/src
wget http://prdownloads.sourceforge.net/tcl/tcl8.5.2-src.tar.gz
wget http://prdownloads.sourceforge.net/tcl/tk8.5.2-src.tar.gz
wget http://prdownloads.sourceforge.net/otcl-tclcl/otcl-src-1.13.tar.gz
tar xzvf tcl8.5.2-src.tar.gz
tar xzvf tk8.5.2-src.tar.gz
tar xzvf otcl-src-1.13.tar.gz
cd /usr/src/tcl8.5.2/unix
./configure
make
make install
cd /usr/src/tk8.5.2/unix
./configure
make
make install
cd /usr/src/otcl-1.13
```

A.1.3 Installation Issues

At this point configure repeatedly failed for *OTcl*, while trying to find the *Tcl* binaries. Both

```
./configure
```

and

```
./configure --with-tcl=/usr/local/bin/tclsh8.5
```

or any other permutation failed.

Version *otcl-1.13* was released 10th March 2007. All of these tools were downloaded and being prepared over a year later on 29th March 2008. The *Tcl/Tk* version 8.5.2 had only been released the day before, on 28th March 2008. It seemed likely that the year-old *OTcl* was incompatible with the new versions of *Tcl* and *Tk*.

It was decided to try reverting to *Tcl/Tk* version 8.4.14, from 19th October 2006 — a version verified to work for current *ns-2* sources. The reversion proceeded as follows.

```
cd /usr/src/tk8.5.2/unix
make distclean
cd /usr/src/tcl8.5.2/unix
make distclean
cd /usr/local/bin
rm -f tclsh* wish*
cd /usr/local/include
rm -f tcl* tk*
cd /usr/local/lib
rm -rf tcl* tk* libtcl* libtk*
cd /usr/local/man/man1
rm -f tclsh* wish*
cd /usr/local/man/man3
rm -f Tcl* Tk* TCL* Ttk*
```

Next, the version 8.4.14 sources for *Tcl* and *Tk* were obtained, compiled and installed, as follows.

```
cd /usr/src
wget http://prdownloads.sourceforge.net/tcl/tcl8.4.14-src.tar.gz
wget http://prdownloads.sourceforge.net/tcl/tk8.4.14-src.tar.gz
tar xzvf tcl8.4.14-src.tar.gz
tar xzvf tk8.4.14-src.tar.gz
```

```
cd /usr/src/tcl8.4.14/unix
./configure
make
make install
cd /usr/src/tk8.4.14/unix
./configure
make
make install
```

Then, *OTcl* version 1.13 was successfully compiled and installed as follows.

```
cd /usr/src/otcl-1.13
./configure
(NOT ./configure --with-tcl=/usr/share/tcl8.4.14/ as in some examples.)
vi Makefile
line 31 change
INST_OLIBSH=    NONE/lib
to
INST_OLIBSH=    /usr/local/lib
make install
```

A.1.4 Download and Install TclCl

Next, a *Tcl/C++* interface, called *TclCL*, for "Tcl with CLasses", was needed to provide the *Tcl/C++* interface for *ns-2* and *nam*. Version 1.18 was used.

```
cd /usr/src
wget http://downloads.sourceforge.net/otcl-tclcl/tclcl-src-1.18.tar.gz
tar xzvf tclcl-src-1.18.tar.gz
cd tclcl-1.18
./configure --with-tcl=/usr/src/tcl8.4.14/
make
make install
```

A.1.5 Download and Install ns-2, nam and xgraph

Finally, *ns-2* and *nam*, along with optional David Harrison's *xgraph*, were built as follows.

```
cd /usr/src
wget http://downloads.sourceforge.net/nsnam/ns-2.33.tar.gz
wget http://downloads.sourceforge.net/nsnam/nam-src-1.13.tar.gz
wget http://downloads.sourceforge.net/nsnam/xgraph-12.1.tar.gz
tar xzvf ns-2.33.tar.gz
tar xzvf nam-src-1.13.tar.gz
tar xzvf xgraph-12.1.tar.gz
cd ns-2.33
./configure
make
make install
echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:' > /etc/p
rofile.d/ns.sh
chmod 0733 /etc/profile.d/ns.sh
```

After the installation, a clean boot with all the new libraries was performed and the simulator was tested as follows.

```
$ ns
% set ns [new Simulator]
_o4
% ^C
```

Then *nam* was built.

```
cd /usr/src/nam-1.13
./configure
make
make install
```

Then check the *nam* console starts.

```
nam
```

Lastly, *xgraph* was built.

```
cd /usr/src/xgraph-12.1
./configure
make
make install
```

Initially, the *Fedora 7* OS was chosen both for its currency over the very stable *CentOS* (a re-compiled community version of RHEL) and its stability over the latest release, at that time, of *Fedora 8* (a *Red Hat* sponsored community *Linux* project). As *ns-2* is compiled against the current running kernel, the frequency of kernel upgrades has considerable impact on research activity involving the simulator. By not being the latest release, at that time, *Fedora 7* was likely to have less frequent kernel upgrades and thus provide a longer mean time between full compilations of the software. However, as the research progressed, *Fedora 7* fell too far behind in development, presenting unsupported packages and a general security risk and so it was decided to move to *Fedora 9*.

A.2 Platform-Specific Issues

Soon after the move to *Fedora 9*, in late August 2008, the *Fedora* software repositories were penetrated and the *Fedora GPG*[1] software signing keys may have been compromised [183]. This prompted the issue of new keys and a new package repository for both *Fedora 8* and *Fedora 9*. *Fedora 7*, being no longer supported, was not included in this process. The possibly compromised keys are listed here.

- RPM-GPG-KEY-fedora (4F2A6FD2),
- RPM-GPG-KEY-fedora-test (30C9ECF8),
- RPM-GPG-KEY-fedora-extras (1AC70CE6), and
- RPM-GPG-KEY-legacy (731002FA).

The key change involved upgrades to the latest *PackageKit* (0.2.5-1.fc9 i386) and a new *fedora-release* (9-5.transition noarch) with new .repo files pointing to the new repositories, but signed with the old key, in the old *updates* repository. This update then loads the new .repo files pointing to the new repositories on old systems accepting the old key, such that any subsequent updates point to the new *updates-newkey* and *updates-testing-newkey* repositories.

```
=============================================================================
 Package                 Arch       Version          Repository        Size
=============================================================================
Updating:
 PackageKit              i386       0.2.5-1.fc9      updates          561 k
 PackageKit-libs         i386       0.2.5-1.fc9      updates          106 k
 fedora-release          noarch     9-5.transition   updates           34 k
```

[1] *GNU Privacy Guard* (*GnuPG* or *GPG*).

```
 gnome-packagekit        i386       0.2.5-2.fc9      updates            1.1 M
 yum-packagekit          i386       0.2.5-1.fc9      updates             11 k

Transaction Summary
=============================================================================
Install      0 Package(s)
Update       5 Package(s)
Remove       0 Package(s)
```

For any case where the original .repo files have been modified, this update leaves two rpmnew .repo files as follows:

```
warning: /etc/yum.repos.d/fedora-updates.repo created as /etc/yum.repos
.d/fedora-updates.repo.rpmnew
warning: /etc/yum.repos.d/fedora.repo created as /etc/yum.repos.d/fedor
a.repo.rpmnew
```

These can then be modified if desired and moved over the active files.

```
mv /etc/yum.repos.d/fedora-updates.repo.rpmnew /etc/yum.repos.d/fedora-
updates.repo
mv /etc/yum.repos.d/fedora.repo.rpmnew /etc/yum.repos.d/fedora.repo
```

However, as all future updates were to come from the *updates-newkey* and *updates-testing-newkey* repositories, this was not required.

The next update then points to the new repositories, signed with the new keys listed here:

- RPM-GPG-KEY-fedora-8-and-9-primary (6DF2196F) and
- RPM-GPG-KEY-fedora-test-8-and-9-primary (DF9B0AE9)

The new keys are then imported as required during the normal update process.

```
warning: rpmts_HdrFromFdno: Header V3 DSA signature: NOKEY, key ID 6df2
196f
Importing GPG key 0x6DF2196F "Fedora (8 and 9) <fedora@fedoraproject.or
g>" from /etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-8-and-9-i386
Is this ok [y/N]:
```

Appendix B

Attacking WPA2

This appendix discusses the results of a series of tests, conducted by the author during the course of this research demonstrating the vulnerabilities from weak configurations of WPA and WPA2 WLANs.

Most of the material here was also presented by the author in "*Securely Deploying IEEE 802.11 WLANs*" [24], at the *AusCERT Asia Pacific Information Technology Security Conference (AusCERT2007)*, Gold Coast, Australia, in May 2007. Some sections of this material were also presented in an Australian Academic and Research Network (AARNet) 'Ozeconference' titled "A Review of Actual IEEE 802.11 Deployment Practices" [11] in May 2008.

The are numerous WPA2-certified devices that are known to permit simultaneous WEP and WPA/WPA2 associations, when configured to do so. It is similarly well known that the majority of commercial-grade equipment is highly-configurable and can be made to accommodate almost anything the administrator desires.

However, there seems to be no empirical data available as to the possibility of common-off-the-shelf devices, such as might be used in small office / home office (SOHO) situations (the consumer-grade devices), allowing simultaneous WEP and WPA2 associations, when configured as "WPA2", presumably intending 'WPA2-only' associations.

Tests were performed on a number of combinations of vendors' devices, with various chipsets.

B.1 Conduct of the Tests

Access Points used were:

- **Linksys** WRT54G "Wireless-G Broadband Router", Wi-Fi Certified "includes WPA2" (as opposed to the Wi-Fi WPA/WPA2 Certifications);
- **Belkin** "Wireless G Router", Wi-Fi Certified and "WPA – WPA2" (but without the Wi-Fi WPA/WPA2 Certification mark); and
- **SMC** SMC2804WBR "Barricade g Wireless Broadband Router", with no Wi-Fi certifications, claiming "Wi-Fi Protected Access (WPA)", but only providing WEP out-of-the-box, with WPA added by a flash upgrade.

Client Stations used were:

- **D-Link** "AirPlus G" DWL-G520 Wireless PCI Adapter (**Atheros** chipset) on **Microsoft** Windows 2000 Professional with D-Link "AirPlus G" DWL-G520 Windows 2000 drivers;
- **D-Link** "AirPlus G" DWL-G520 Wireless PCI Adapter (**Atheros** chipset) on **Fedora** with **madwifi** drivers and **wpa_supplicant**;
- **D-Link** "AirPlus G+" DWL-G520+ Wireless PCI Adapter (**Texas Instruments** ACX110 chipset) on **Microsoft** Windows XP Professional with D-Link "AirPlus G+" DWL-G520+ Windows XP drivers; and
- **D-Link** "AirPlus G+" DWL-G520+ Wireless PCI Adapter (**Texas Instruments** ACX110 chipset) on **Fedora** with **ndiswrapper** and D-Link "AirPlus G+" DWL-G520+ Windows XP drivers.

All data capture was performed on a separate machine with a **D-Link** "AirPlus G" DWL-G650 Wireless PCMCIA Adapter (**Atheros** chipset) on **Fedora** with **madwifi** drivers, **kismet** and **ethereal**[1].

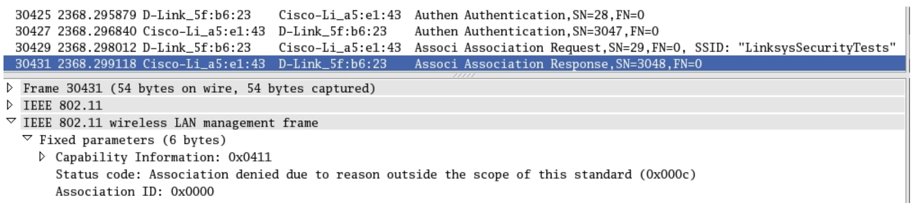
B.1.1 Initial Tests

The initial tests were performed with the **Microsoft** Windows clients to the **Linksys** WRT54G on a "*LinksysSecurityTests*" network. The WRT54G allowed the selection of WPA2 with AES or TKIP+AES or WPA with AES or TKIP or

[1]The Ethereal Project now uses the name "Wireshark".

WEP, with all possible combinations of authentication and key-modes. The Windows clients offered a simplified range of options with WPA2, WPA2-Personal, WPA, WPA-Personal and various WEP flavours.

All legal combinations of authentication and encryption were tested and the clients were able to connect for all matching combinations and, as required and expected, mixed encryption modes could not connect. These typically were indicated by the client STA finding no suitable AP (with intersecting sets of protocols) in the responses to its directed and broadcast probes. When Linux clients were used, so that association could be forced by the client STA, the AP would typically respond with the association denied for an unknown reason.

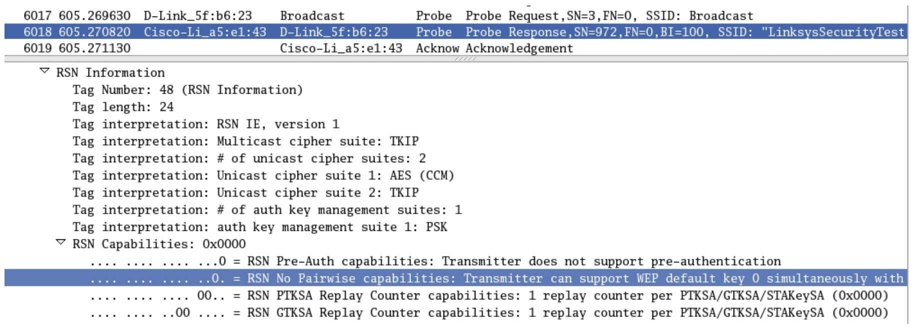


```
30425 2368.295879 D-Link_5f:b6:23    Cisco-Li_a5:e1:43  Authen Authentication,SN=28,FN=0
30427 2368.296840 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Authen Authentication,SN=3047,FN=0
30429 2368.298012 D-Link_5f:b6:23    Cisco-Li_a5:e1:43  Associ Association Request,SN=29,FN=0, SSID: "LinksysSecurityTests"
30431 2368.299118 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Associ Association Response,SN=3048,FN=0

▷ Frame 30431 (54 bytes on wire, 54 bytes captured)
▷ IEEE 802.11
▽ IEEE 802.11 wireless LAN management frame
  ▽ Fixed parameters (6 bytes)
    ▷ Capability Information: 0x0411
      Status code: Association denied due to reason outside the scope of this standard (0x000c)
      Association ID: 0x0000
```

Figure B.1: AP Association Response with mismatched protocols

However, a WEP client attempting to associate with the WRT54G AP set for WPA2 (only), but with both TKIP and CCMP and the default WEP Key 0 set to match, revealed what appeared to be the AP offering the possibility of a WEP connection:



```
6017 605.269630  D-Link_5f:b6:23    Broadcast          Probe  Probe Request,SN=3,FN=0, SSID: Broadcast
6018 605.270820  Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Probe  Probe Response,SN=972,FN=0,BI=100, SSID: "LinksysSecurityTest
6019 605.271130                     Cisco-Li_a5:e1:43  Acknow Acknowledgement

    ▽ RSN Information
        Tag Number: 48 (RSN Information)
        Tag length: 24
        Tag interpretation: RSN IE, version 1
        Tag interpretation: Multicast cipher suite: TKIP
        Tag interpretation: # of unicast cipher suites: 2
        Tag interpretation: Unicast cipher suite 1: AES (CCM)
        Tag interpretation: Unicast cipher suite 2: TKIP
        Tag interpretation: # of auth key management suites: 1
        Tag interpretation: auth key management suite 1: PSK
      ▽ RSN Capabilities: 0x0000
          .... .... .... ...0 = RSN Pre-Auth capabilities: Transmitter does not support pre-authentication
          .... .... .... ..0. = RSN No Pairwise capabilities: Transmitter can support WEP default key 0 simultaneously with
          .... .... .... 00.. = RSN PTKSA Replay Counter capabilities: 1 replay counter per PTKSA/GTKSA/STAKeySA (0x0000)
          .... .... ..00 .... = RSN GTKSA Replay Counter capabilities: 1 replay counter per PTKSA/GTKSA/STAKeySA (0x0000)
```

Figure B.2: RSN Transmitter can support WEP simultaneously

This was also noted in IEEE 802.11i, verbatim:

> “Bit 1: No Pairwise. If a STA can support WEP default key 0 simultaneously with a pairwise key.” [12, Para 7.3.2.25.3]

This looked promising for a attempt to get a WEP association with a WPA2 AP, to compromise the WEP key and thereby subvert the security of the underlying network.

B.1.2 A False Positive Result

As expected, the Windows clients initially were unable to associate, via WEP, with the WRT54G configured for WPA2 "TKIP+AES" mode. However, at one point, after a series of configuration changes, the **Microsoft** Windows 2000 Professional with D-Link "AirPlus G" DWL-G520 client indicated it had associated with the **Linksys** WRT54G via WEP. A valid association was verified by pinging a host behind the AP via the WLAN.

The next phase was to gather enough packets to execute the KoreK attacks and recover the WEP key. Traffic was simulated with a continuous ping (one per second) from the Windows 2000 host. Unfortunately, after a period of time, the connection would fail and the drivers lock-up, requiring a reboot. This apparent instability was severely hampering the gathering of packets for the statistical attack and it was decided to switch to a Linux client with a more versatile set of network tools and possibly less stability problems.

Two different Linux clients with different hardware and software were used. One used the **D-Link** "AirPlus G+" DWL-G520+, with a **Texas Instruments** ACX110 chipset and **ndiswrapper** and the D-Link Windows XP drivers — chosen because of the same binary drivers as the Windows clients, so as to differentiate any OS-specific issues.

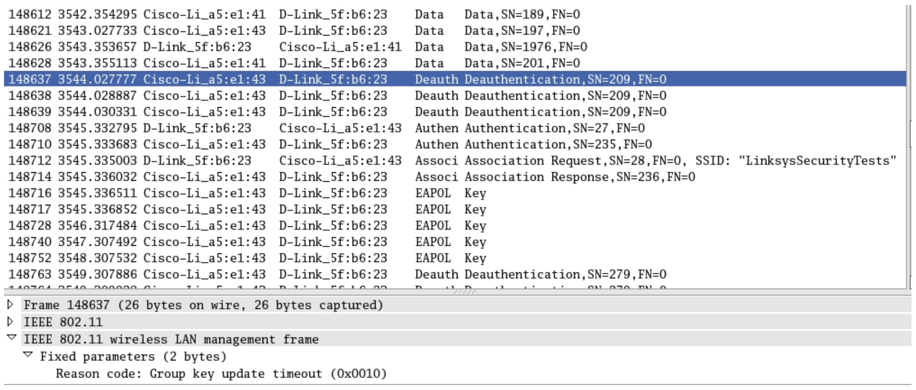
The other used the **D-Link** "AirPlus G" DWL-G520, with an entirely different **Atheros** chipset and **madwifi** drivers and **wpa_supplicant**. This combination was chosen because the **Atheros** chipset is highly configurable, performing much of its work in software and firmware, rather than dedicated hardware. The **madwifi** drivers give an extensive interface to this chipset and the **wpa_supplicant** uses a detailed configuration file, allowing all settings to be manipulated.

Extensive testing involving all possible configurations of the clients and the WRT54G as WPA or WPA2 failed to obtain a WEP association. The only recourse was to revert to the Windows client and gather the necessary packets over a longer time. However, now the Windows client also refused to associate via WEP. There now seemed only two possibilities, either:

- The original WEP associations had actually been with some other AP, allowing WEP; or
- The original associations had not actually been WEP or the AP was allowing WEP.

B.1.3 Analysis

The packet capture files for the month in question were reviewed to find that the client had been authenticating and associating with the correct AP, the WRT54G, but even though the Windows client had consistently reported WEP encryption in use, the traffic clearly showed a TKIP protocol selection and 4-way handshake for a WPA connection and the apparent 'instability' was in fact due to the WEP key being used for the group cipher and the disassociate in group key time-out never recovering (because the client will not negotiate a new key) and the new handshake timing out.

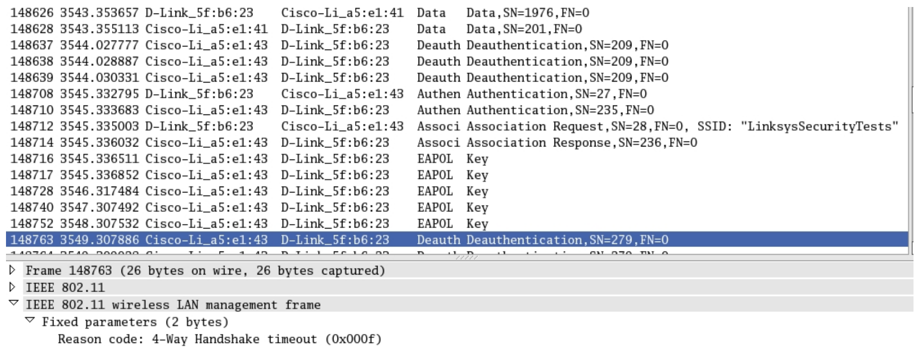


```
148612 3542.354295 Cisco-Li_a5:e1:41  D-Link_5f:b6:23    Data   Data,SN=189,FN=0
148621 3543.027733 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Data   Data,SN=197,FN=0
148626 3543.353657 D-Link_5f:b6:23    Cisco-Li_a5:e1:41  Data   Data,SN=1976,FN=0
148628 3543.355113 Cisco-Li_a5:e1:41  D-Link_5f:b6:23    Data   Data,SN=201,FN=0
148637 3544.027777 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Deauth Deauthentication,SN=209,FN=0
148638 3544.028887 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Deauth Deauthentication,SN=209,FN=0
148639 3544.030331 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Deauth Deauthentication,SN=209,FN=0
148708 3545.332795 D-Link_5f:b6:23    Cisco-Li_a5:e1:43  Authen Authentication,SN=27,FN=0
148710 3545.333683 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Authen Authentication,SN=235,FN=0
148712 3545.335003 D-Link_5f:b6:23    Cisco-Li_a5:e1:43  Associ Association Request,SN=28,FN=0, SSID: "LinksysSecurityTests"
148714 3545.336032 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Associ Association Response,SN=236,FN=0
148716 3545.336511 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    EAPOL  Key
148717 3545.336852 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    EAPOL  Key
148728 3546.317484 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    EAPOL  Key
148740 3547.307492 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    EAPOL  Key
148752 3548.307532 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    EAPOL  Key
148763 3549.307886 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Deauth Deauthentication,SN=279,FN=0

▷ Frame 148637 (26 bytes on wire, 26 bytes captured)
▷ IEEE 802.11
▽ IEEE 802.11 wireless LAN management frame
  ▽ Fixed parameters (2 bytes)
      Reason code: Group key update timeout (0x0010)
```

Figure B.3: Disassociate on Group Key Time-out

Further investigation revealed the 'No Pairwise' bit **only has meaning for a client STA**, to indicate to an AP as to whether or not it can support WEP default key 0 simultaneously with a pairwise key. (So that the AP does not attempt to have the STA install a pairwise key, over the top of WEP default key 0.) This bit has no relevance in packets from an AP. An AP **always** sets this bit to zero. While this is not new, it was, for this author, obscure, and is included for emphasis.



```
148626 3543.353657 D-Link_5f:b6:23    Cisco-Li_a5:e1:41  Data   Data,SN=1976,FN=0
148628 3543.355113 Cisco-Li_a5:e1:41  D-Link_5f:b6:23    Data   Data,SN=201,FN=0
148637 3544.027777 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Deauth Deauthentication,SN=209,FN=0
148638 3544.028887 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Deauth Deauthentication,SN=209,FN=0
148639 3544.030331 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Deauth Deauthentication,SN=209,FN=0
148708 3545.332795 D-Link_5f:b6:23    Cisco-Li_a5:e1:43  Authen Authentication,SN=27,FN=0
148710 3545.333683 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Authen Authentication,SN=235,FN=0
148712 3545.335003 D-Link_5f:b6:23    Cisco-Li_a5:e1:43  Associ Association Request,SN=28,FN=0, SSID: "LinksysSecurityTests"
148714 3545.336032 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Associ Association Response,SN=236,FN=0
148716 3545.336511 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    EAPOL  Key
148717 3545.336852 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    EAPOL  Key
148728 3546.317484 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    EAPOL  Key
148740 3547.307492 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    EAPOL  Key
148752 3548.307532 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    EAPOL  Key
148763 3549.307886 Cisco-Li_a5:e1:43  D-Link_5f:b6:23    Deauth Deauthentication,SN=279,FN=0
▷ Frame 148763 (26 bytes on wire, 26 bytes captured)
▷ IEEE 802.11
▽ IEEE 802.11 wireless LAN management frame
  ▽ Fixed parameters (2 bytes)
      Reason code: 4-Way Handshake timeout (0x000f)
```

Figure B.4: Subsequently fails to negotiate a new Group Key

B.1.4 Remaining Results

With this avenue of attack failing to be realised, the tests were repeated on the other APs.

None of the variations of clients, for either **Atheros** or **Texas Instruments** chipsets, combined with either **Microsoft** or **Linux** driver software, on any of the platforms, could successfully associate with WEP with any of the APs surveyed when correctly configured for WPA2 (only).

B.1.5 Weak Configurations

Neither the **Linksys** WRT54G, nor the **Belkin** "Wireless G Router" provided a configuration option permitting the use of WEP with WPA modes.

The **SMC** SMC2804WBR "Barricade g Wireless Broadband Router" does offer a WEP/WPA combined mode. With this mode set both WPA and WEP associations can exist simultaneously.

The SMC2804WBR was configured for WEP/WPA combined mode and simultaneous associations were successfully made with:

- the **D-Link** "AirPlus G+" DWL-G520+ Wireless PCI Adapter (**Texas Instruments** ACX110 chipset) on **Microsoft** Windows XP Professional with D-Link "AirPlus G+" DWL-G520+ Windows XP drivers configured for WPA; and
- the **D-Link** "AirPlus G" DWL-G520 Wireless PCI Adapter (**Atheros** chipset) on **Fedora** with **madwifi** drivers and **wpa_supplicant** configured for WEP.

Both the WPA and WEP clients were able to access the network behind the SMC2804WBR, confirmed by pinging hosts on the wired network.

By setting continuous traffic from the WEP client, it was a trivial matter to capture the traffic needed to use with **aircrack** to recover the WEP key on the separate "intruder" device, a laptop with a **D-Link** "AirPlus G" DWL-G650 Wireless PCMCIA Adapter (**Atheros** chipset) on **Fedora** with **madwifi** drivers, **kismet**, **ethereal** (now "Wireshark") and **aircrack**.

It was found that the WEP key allowed an attacker to get to the entire wired LAN behind the SMC2804WBR, severely compromising the wired LAN, including access to every host NIC (although host security provided protection) and by setting the SMC2804WBR as the default gateway, all other subnets and the Internet service. This required the SMC2804WBR to be configured as a gateway, allowing any out, with the appropriate default routes.

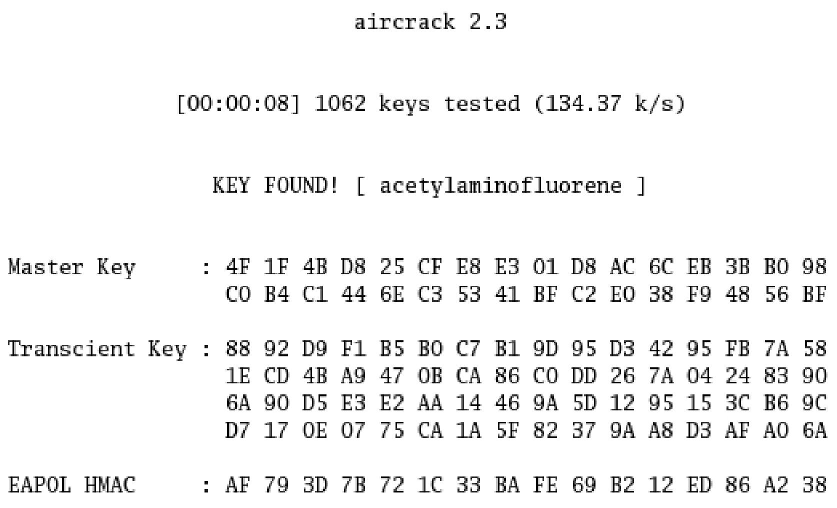
Curiously, although the WEP key allowed an attacker to get to the entire wired LAN, we were not able to access the WPA devices using the same AP for the same wired LAN.

Again, this is not a clandestine attack — the SMC2804WBR has to be configured as WEP/WPA to allow this attack. This is a clearly dangerous configuration.

B.1.6 Using Pre-Shared Keys with WPA and WPA2

Note that in the simplest case, for small private networks, there is the option of using a Pre-Shared Key (PSK) with WPA2 (WPA2-PSK). The Wi-Fi Alliance calls this "WPA2-Personal" (as opposed to "WPA2-Enterprise" that uses the 802.1X authentication methods).

The remaining issue, which is also shown in the discussion in the next section, was the use of 'Pre-Shared Key' (PSK) mode in all of these tests. It took **aircrack** only eight seconds to find our reasonably long (19 characters and 20 characters), but very poorly chosen pass-phrases, "acetylaminofluorene" and "acetylcholinesterase" that were used for WPA and WPA2 key pass-phrases throughout these tests.



```
                        aircrack 2.3

             [00:00:08] 1062 keys tested (134.37 k/s)

                KEY FOUND! [ acetylaminofluorene ]

Master Key     : 4F 1F 4B D8 25 CF E8 E3 01 D8 AC 6C EB 3B B0 98
                 C0 B4 C1 44 6E C3 53 41 BF C2 E0 38 F9 48 56 BF

Transcient Key : 88 92 D9 F1 B5 B0 C7 B1 9D 95 D3 42 95 FB 7A 58
                 1E CD 4B A9 47 0B CA 86 C0 DD 26 7A 04 24 83 90
                 6A 90 D5 E3 E2 AA 14 46 9A 5D 12 95 15 3C B6 9C
                 D7 17 0E 07 75 CA 1A 5F 82 37 9A A8 D3 AF A0 6A

EAPOL HMAC     : AF 79 3D 7B 72 1C 33 BA FE 69 B2 12 ED 86 A2 38
```

Figure B.5: Pre-Shared Key (PSK) broken in 8 seconds [24]

B.2 Discussion

Some device drivers allow a PSK to be entered as 64 hexadecimal digits providing the 256-bit key directly, such as (often with a leading ‘0x’)
‘8F8E17FC4CFD675A4F70E38587843845097A6D34637CA4F71FF785EBB462D71E’.
However, unless **every** device on the network allows this method of entry, it can not be used for any of the devices, as the other entry methods hash the input.

The more universal method of setting the PSK is to supply a 8–63[2] character pass-phrase. The IEEE 802.11i standard uses the term “pass-phrase”, meaning a secret text string, to distinguish it from a ‘password’ commonly used to mean a single group of alphabetic symbols containing no spaces. There is a very great risk to the security of the configuration if the pass-phrase used has insufficient entropy, making it susceptible to dictionary attacks. Ideally the pass-phrase should be 63 random printable characters, such as
‘Fi+Kbl).1x@/X5Lf8sNf;AYl[!eCU:’y72aSBoB(=P4k=GzLna:F*bKJ?.gAwq@’.
However some vendor’s interfaces cannot accept the full encoding range of 32 to 126 (decimal), inclusive and the pass-phrase is restricted to alphanumeric characters, such as
‘L7YavIR03qUQbrVWexMqDrGSK8gCxdjATZVSllFiTpPZUzSoPjrofrxokoV6cih’.

A common comment in Internet forums is that the pass-phrase should be at least 20 characters long. However this should be random characters, or there will be insufficient entropy. Many make the mistake of using an actual *phrase*, or worse a single pass*word*. Figure B.5 shows how quickly tools like ***aircrack*** can break non-random words with a dictionary attack — in this case, eight seconds.

B.2.1 General Guidelines for Implementing a RSN

Since any implementation of a RSN on WPA2 equipment demands the use of AES-CCMP only for both the (singular) pairwise suite, as well as the groupwise suite, the following guidelines apply equally to public, private and commercial WLANs:

1. All of the devices must implement CCMP (i.e. must all be WPA2 devices).

2. All of the devices must not permit pre-RSNAs:

    - Select “WPA2 Only Mode”

[2]To distinguish it from the 64 digit hex.

  - Ensure the transmitted RSN IE uses only CCMP for the groupwise suite
  - Test to ensure WEP STA is refused association — in both *shared-key* and *open* authentication modes

3. If the software/firmware allows, remove or disable WEP-40, WEP-104 and TKIP.

4. If using WPA2-PSK, use a random 256-bit (64 hexadecimal digit) key.

5. If using PSK, but cannot use hex keys, use a random 63 character 'pass-phrase'.

6. If using 802.1X authentication, ensure the selected EAP provides **mutual** authentication, such as EAP-TLS.

7. If the software/firmware allows, remove or disable other EAPs.

There is a critical difference between *RSNA-capable* devices and *RSNA-enabled* devices. It has been shown that conforming with the IEEE 802.11i amendment does not necessarily enforce a RSN and the WPA2 certification of equipment is not sufficient to provide a RSN, since compliance with IEEE 802.11i does not mandate a RSN in operation.

WPA2 certified equipment will implement a RSN, if properly configured to do so, but this may not be the default. With WPA2 certification requiring backward compatibility with WPA, it is likely that WPA2 devices will default to a TSN. In addition to this, many vendors provide various other mixed pre-RSNA modes of operation.

Bibliography

[1] James Clerk Maxwell. A Dynamical Theory of the Electromagnetic Field. *Philosophical Transactions of the Royal Society of London*, (155):459–512, 1865.

[2] Tapan K. Sarkar, Magdalena Salazar-Palma, and Dipak Sengupta. A Chronology of Developments of Wireless Communication and Electronics from 1831–1920. In *Proc. Antennas and Propagation Society International Symposium, 2001*, volume 1, pages 2–5. IEEE, 8–13 July 2001.

[3] William E. Gordon. A Hundred Years of Radio Propagation. *IEEE Transactions on Antennas and Propagation*, 33(2):126–130, February 1985.

[4] Probir K. Bondyopadhyay. Comments on 'A Hundred Years of Radio Propagation'. *IEEE Transactions on Antennas and Propagation*, 38(10):1723–1726, October 1990.

[5] IEEE Std 802.11–1997. *Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11:Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 1997. Approved 26 June 1997.

[6] David Conran. WarBussing: The State of Wireless Security in a cross section of a major Australian City. In *Proc. AusCERT Asia Pacific Information Technology Security Conference 2005*. Australian Computer Emergency Response Team, May 2005.

[7] Monash University Information Technology Services. Monash Student use of IT. Summary of 'The way students use IT' symposium, 8th Decem-

ber 2006. [Online] Available: `www.calt.monash.edu.au/Quality/ETC/agenda/content/2006-exec-summary-students-it-final.doc`

[8] Queensland Government Chief Information Office. 2006 Queensland Household Survey — Computer and Internet Usage, 2006. Survey conducted by the Office of the Economic and Statistical Research (OESR) from Monday 10 May to Thursday 10 June 2006. [Online] Available: `www.qgcio.qld.gov.au/00_pdf/hhold_report_6.pdf`

[9] ISO/IEC 8802-11: 1999(E); ANSI/IEEE Std 802.11, 1999 Edition. *Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 20th August 1999. Approved 18 March 1999.

[10] Patryk Szewczyk. Individuals Perceptions of Wireless Security in the Home Environment. In Dr. Craig Valli and Dr. Andrew Woodward, editors, *Proceedings of 4th Australian Information Security Management Conference*, Perth, Western Australia, December 2006. School of Computer and Information Science, Edith Cowan University. ISBN 0-7298-0625-1.

[11] David Ross. A Review of Actual IEEE 802.11 Deployment Practices. AARNet Ozeconference, 14th May 2008. Australian Academic and Research Network. [Online] Available: `http://www.aarnet.com.au/Events/2008/05/02/Ozeconf--Wireless-Security.aspx`

[12] IEEE Std 802.11i–2004. *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 6: Medium Access Control (MAC) Security Enhancements.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 23rd July 2004. [Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003) as amended by IEEE Stds 802.11a–1999, 802.11b–1999, 802.11b–1999/Cor 1–2001, 802.11d–2001, 802.11g–2003, and 802.11h–2003].

[13] David B. Nelson and Kevin N. Hayes. IEEE P802.11 Wireless LANs Letter Ballot 64 Comment Resolution Motions. doc.: IEEE 802.11-04/088r0, January 2004. Dated 11th November 2003. [Online] Available: `https://mentor.ieee.org/802.11/dcn/04/11-04-0088-00-000i-lb64-comment-resolution-motions.doc`

[14] Arshad Aziz and Nassar Ikram. Hardware implementation of AES-CCM for robust secure wireless network. In Jan Eloff, Hein Venter, Les Labuschagne, and Mariki Eloff, editors, *Peer-reviewed Proceedings of the ISSA 2005 New Knowledge Today Conference*, Balalaika Hotel, Sandton, South Africa, 29th June – 1st July 2005. Information Security South Africa (ISSA). ISBN 1-86854-625-X.

[15] Abdul Samiah, Arshad Aziz, and Nassar Ikram. An Efficient Software Implementation of AES-CCM for IEEE 802.11i Wireless Standard. In *Proc. 31st Annual International Computer Software and Applications Conference(COMPSAC 2007)*, volume 2, pages 689–694, Beijing, China, 24–27 July 2007. The Institute of Electrical and Electronics Engineers, Inc., IEEE Computer Society Conference Publishing Services (CPS). ISBN: 0-7695-2870-8.

[16] David Johnston. AES-CCM Encryption and Authentication Mode for 802.16. IEEE 802.16 Broadband Wireless Access Working Group C802.16e-04/12, 10th January 2004.

[17] Kwang-Ok Kim, Kyeong-Soo Han, and Tae-Whan Yoo. The Implementation of the Link Security Module in an EPON Access Network. In *Proc. IEEE 11th Asia Pacific Conference on Communications (APCC 2005)*, pages 1026–, Perth, Western Australia, 3rd–5th October 2005. Electronics and Telecommunications Research Institute, Daejeon, Korea.

[18] Kyeong-Soo Han, Kwang-Ok Kim, Tae Whan Yoo, and Yul Kwon. The Design and Implementation of MAC Security in EPON. In *Proc. The 8th International Conference on Advanced Communication Technology, 2006. ICACT 2006.*, pages 1673–1676, Phoenix Park, Korea, 20-22 February 2006. Electronics and Telecommunications Research Institute, Daejeon, Korea. ISBN 89-5519-129-4.

[19] Allyn Romanow. IEEE 802.1 Link Security Study Group, June 2003 meeting minutes, 2nd–3rd June 2003. Dolors Sala, chair.

[20] IEEE Std 802.1AE–2006. *IEEE Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Security.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 18th June 2006. ISBN: 0-7381-4990-X.

[21] Prashant Dewan, Larry Swanson, and Men Long. Are Your Company Secrets Really Safe? Intel Research vendor white paper on LinkSec Layer 2 Security, 18th December 2006. Intel Corporation. [Online] Available: `http://www.intel.com/netcomms/casestudies/linksec.pdf`

[22] Zhen-Ming Yang and Guo-Zhu Liu. Implementing Romote[sic] Register of Software by NC MAC Adress IN C#. *Qingdao University of Science and Technology Journal (Natural Science)*, 26(3):259–263, June 2005. English abstract from Chinese text, Chinese Electronic Periodical Services, Honor digital Inc. [Online] Available: `http://www.ceps.com.tw/ec/ecjnlarticleView.aspx?atliid=161366&issueiid=11379&jnliid=1194`

[23] David Ross. The Security of Wireless Computing Technologies. In Andrew Clark, Kathryn Kerr, and George Mohay, editors, *Proc. AusCERT Asia Pacific Information Technology Security Conference Refereed R&D Stream*, pages 51–63. Australian Computer Emergency Response Team, May 2005. ISBN: 1-86499-799-0.

[24] David Ross, Andrew Clark, and Mark Looi. Securely Deploying IEEE 802.11 WLANs. In A. Clark, M. McPherson, and G. Mohay, editors, *Proceedings of AusCERT Asia Pacific Information Technology Security Conference (AusCERT2007): Refereed R&D Stream*, pages 50–70. Australian Computer Emergency Response Team, University of Queensland, May 2007. ISBN: 978-1-86499-877-1.

[25] The CMU Monarch Project. The CMU Monarch Projects Wireless and Mobility Extensions to ns. Snapshot Release 1.1.1, 5th August 1999. Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA.

[26] Bruce Moore, editor. *The Australian Oxford Dictionary.* Australian National Dictionary Centre, Canberra, Australia, second edition, 2004. ISBN 0 19 551796 2.

[27] Robin Fairbairns. The UK TEX FAQ Your 244 Questions Answered, 29th May 2002. [Online] Available: `http://www.ntg.nl/doc/fairbairns/newfaq.pdf`

[28] IEEE. *IEEE Editorial Style Manual*, (undated). [Online] Available: `http://www.ieee.org/portal/cms_docs_iportals/iportals/publications/authors/transjnl/stylemanual.pdf`

[29] Michael Shell. *How to Use the IEEEtran BibTEXStyle*, 20th June 2002. revised September 30, 2008. [Online] Available: `ftp://tug.ctan.org/pub/tex-archive/macros/latex/contrib/IEEEtran/bibtex/IEEEtran_bst_HOWTO.pdf`

[30] Bureau International des Poids et Mesures. *The International System of Units (SI)*, 8th edition, March 2006. Organisation Intergouvernementale de la Convention du Mètre (The International Bureau of Weights and Measures (BIPM) was set up by the Metre Convention signed in Paris on 20 May 1875.).

[31] David Ross. The Security of Wireless Computing Technologies. In Andrew Clark, Kathryn Kerr, and George Mohay, editors, *Proc. AusCERT Asia Pacific Information Technology Security Conference Refereed R&D Stream*, May 2005. Original abstract published on the conference website, later changed before publication in proceedings. [Online] Available: `conference.auscert.org.au/conf2005/abstracts.php`

[32] John R. Platt. IEEE 802.11: From Blackberrys To Bunnies. *the institute*, 31(2):6, June 2007. IEEE.

[33] Yasir Zahur and T. Andrew Yang. Wireless LAN security and laboratory designs. *Journal of Computing in Small Colleges*, 19(3):44–60, January 2004. The Consortium for Computing in Small Colleges, USA: new names The Consortium for Computing Sciences in Colleges and Journal of Computing Sciences in Colleges.

[34] Stephen Lawson () 02/03/2009. Alaska becomes latest airline with Wi-Fi. IDG News Service, accessed via Techworld, 2nd March 2009. International Data Group, Inc. [Online] Available: `https://www.techworld.com.au/article/278380/alaska_becomes_latest_airline_wi-fi`

[35] David Ross. Top Reasons for Successful WLAN Penetrations in the Past Year. In *Proc. 16th Australian System Administrators Conference (SAGE-AU'2008)*, Adelaide, Australia, August 2008. The System Administrators Guild of Australia (SAGE-AU).

[36] Cameron Stewart. Parliament House fights hackers. Australian IT, in *The Australian* newspaper, 28th November 2007. News Limited. [Online] Available: `http://www.australianit.news.com.au/story/0,25197,22835163-15306,00.html`

[37] Australian Associated Press Pty Limited (AAP). Parliament House goes wireless to improve security. ZDNet Australia, 28th November 2007. CBS Interactive, a CBS Company. [Online] Available: `http://www.zdnet.com.au/news/security/soa/Parliament-House-goes-wireless-to-improve-security/0,130061744,339284121,00.htm`

[38] John Cox. Report forecasts WLAN 'last-mile' boom. Network World Fusion, 5th August 2002. Network World, Inc. [Online] Available: `http://www.nwfusion.com/news/2002/0805alex.html`

[39] Doug Ramsey. Disaster Drill Tests New Wireless Technologies Developed At UCSD And Cal-(IT). University of California, San Diego, 18th May 2004. Regents of the University of California. [Online] Available: `http://ucsdnews.ucsd.edu/newsrel/general/05_18_WIISARD.asp`

[40] Ellen Romer. U.S. Closing Mobile Usage Gap. white paper, April 2008. Experian Consumer Research. [Online] Available: `http://www.smrb.com/uploads/MobileWP_06.06.08.pdf`

[41] Infrared Data Association. About IrDA, 18th May 2004. [Online] Available: `http://www.irda.org/`

[42] Vishay Semiconductors. *Infrared Data Communication According to IrDA Standard*, rev. 1.4 edition, 20th September 2006. Document number: 82513. [Online] Available: http://www.vishay.com/docs/82513/physical.pdf

[43] Dan Reain. Security in Wireless Mobile Communications. As part of GIAC practical repository, 10th January 2004. SANS Institute. [Online] Available: http://www.giac.org/practical/GSEC/Dan_Reain_GSEC.pdf

[44] Lee Garber. Will 3G Really Be the Next Big Wireless Technology. *Computer*, 35(1):26–32, January 2002. IEEE Computer Society.

[45] Bluetooth SIG. *Specification of the Bluetooth System*. Bluetooth SIG, Inc., version 2.0 + edr edition, 4th November 2004.

[46] Patrick Mannion. Bluetooth road map ups data rates, lowers power. EE Times, 8th November 2004. CMP Media LLC. [Online] Available: http://www.eetimes.com/issue/mn/showArticle.jhtml?articleId=52200120

[47] Jim Zyren. Reliability of IEEE 802.11 Hi Rate DSSS WLANs in a High Density Bluetooth Environment. Technical report, Harris Semiconductor — Communications Products, 8th June 1999.

[48] Glenn Derene. Wireless Made Simple. Newsweek, Inc, 10th May 2004. [Online] Available: http://msnbc.msn.com/id/4945049/

[49] MobileInfo.com. Bluetooth Air Interface and Frequency Band, 28th July 2002. [Online] Available: http://www.mobileinfo.com/Bluetooth/air_&_band.htm

[50] Cheng Lerong. The bluetooth standard, 15th October 2004. [Online] Available: http://www.ee.ucla.edu/~lerong/ee202a/hw2/

[51] IEEE Std 802.15.1–2002. *IEEE Standard for information technology — Telecommunication and information exchange between systems — LAN/MAN — Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPANs)*. IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 14th June 2002.

[52] IEEE Std 802.15.2–2003. *IEEE Recommended Practice for Information Technology—Part 15.2: Coexistence of Wireless Personal Area Networks with Other Wireless Devices Operating in Unlicensed Frequency Bands.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 2003.

[53] Damien O'Rourke. Privacy Concerns for UWB technology? Public posting to the Cryptography mailing list, 2nd April 2004. (cryptography@metzdowd.com). [Online] Available: http://www.mail-archive.com/cryptography@metzdowd.com/msg01875.html

[54] Matt Hamblen. Ultrawideband: A Better Bluetooth. Computerworld, 2nd August 2004. International Data Group, Inc. [Online] Available: http://www.computerworld.com/mobiletopics/mobile/technology/story/0,10801,94896,00.html

[55] Eric S. Brown. Bluetoothful Loser. Technology Review, 15th July 2004. Massachusetts Institute of Technology. [Online] Available: http://www.technologyreview.com/articles/04/07/wo_brown071504.asp

[56] Bob Heile. Re: withdrawal of the 802.15.3a PAR. Letter to the IEEE-SA New Standards Committee (NesCom), 23rd February 2006. [Online] Available: http://standards.ieee.org/board/nes/projects/802-15-3a.pdf

[57] Rick Alfvin. IEEE 802.15 WPAN High Rate Alternative PHY Task Group 3a (TG3a) Home Page, 2006. [Online] Available: http://www.ieee802.org/15/pub/TG3a.html

[58] Federal Communications Commission. *First Report and Order In the matter of Revision of Part 15 of the Commissions Rules Regarding Ultra-Wideband Transmission Systems (FCC 02-48)*, 22nd April 2002. ET Docket 98-153. [Online] Available: http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-02-48A1.pdf

[59] Australian Communications Authority. Use of ultra wideband approved for the first time. ACA Media release No. 24, 1st April 2004. The Australian Communications Authority is now the Australian Communications and Media Authority and aca.gov.au URLS must now be accessed via the

acma.gov.au website. [Online] Available: http://www.acma.gov.au/WEB/STANDARD/pc=PC_1038

[60] Australian Communications and Media Authority. Licensing for anti-collision vehicle radar. ACMA media release 80/2006, 2nd August 2006. [Online] Available: http://www.acma.gov.au/WEB/STANDARD/pc=PC_100684

[61] Sinem Coleri Ergen. ZigBee/IEEE 802.15.4 Summary. Technical report, Advanced Technology Lab of National Semiconductor, 10th September 2004. [Online] Available: http://www.sinemergen.com/zigbee.pdf

[62] Rick Alfvin. IEEE 802.15 WPAN Task Group 5 (TG5) Mesh Networking, 4th April 2009. [Online] Available: http://ieee802.org/15/pub/TG5.html

[63] IEEE Std 802.16–2004. *IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 2004. Revision of IEEE Std 802.16–2001.

[64] Michael Finneran. WiMax Hits The Road. Business Communications Review, pp. 30-37, June 2004. MediaLive International, Inc. [Online] Available: http://www.bcr.com/bcrmag/2004/06/p30.php

[65] Eric Griffith. WECA becomes Wi-Fi Alliance. Wi-Fi Planet, 2nd October 2002. Jupitermedia Corporation. [Online] Available: http://www.wi-fiplanet.com/news/article.php/1474361

[66] John Vollbrecht, David Rago, and Robert Moskowitz. *Wireless LAN Access Control and Authentication.* Interlink Networks Inc., TruSecure Corporation, 2001. Vendor white paper from LeapPoint Technologies via SecurityPeerPublishing. [Online] Available: security.ittoolbox.com/browse.asp?c=SecurityPeerPublishing&r=\%2Fpub\%2FLP073002.pdf

[67] Interlink Networks, Inc. *Wireless LAN Security using Interlink Networks Secure.XS Software and Cisco LEAP*, 2002. Application Notes at Interlink Networks Resource Library. [Online] Available: www.interlinknetworks.com/images/products/Wireless_LAN_Security_SecureXS_and_LEAP.pdf

[68] Australian Communications Authority. WLANS Interference Management, July 2002. [Online] Available: `http://www.aca.gov.au/radcomm/frequency_planning/radiofrequency_planning_topics/docs/rlan-im.pdf`

[69] Australian Communications Authority. *Australian Radiofrequency Spectrum Plan*, January 2002. [Online] Available: `http://www.aca.gov.au/radcomm/frequency_planning/spectrum_plan/arsp02.pdf`

[70] Michael Vollmer. Physics of the Microwave Oven. *Physics Education*, 39(1):74–81, January 2004.

[71] ISO/IEC 8802-11:1999/Amd 1:2000; IEEE Std 802.11a–1999. *Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHZ Band.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 30th December 1999. (Supplement to ANSI/IEEE Std 802.11–1999).

[72] Australian Communications Authority. Proposals for Spectrum Arrangements for RLANs and FWA Systems in the 5 GHz Frequency Range post WRC-03 — a discussion paper, December 2003. [Online] Available: `http://www.aca.gov.au/radcomm/frequency_planning/spps/0313spp.pdf`

[73] Cisco Systems, Inc. Cisco Aironet Antenna Reference Guide, 25th November 2004. [Online] Available: `http://www.cisco.com/warp/public/cc/pd/witc/ao350ap/prodlit/agder_rg.pdf`

[74] IEEE Std 802.11b–1999. *Supplement to IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 20th January 2000. (Supplement to ANSI/IEEE Std 802.11, 1999 Edition).

[75] IEEE Std 802.11g–2003. *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 25th June 2003. [Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003) as amended by IEEE Stds 802.11a–1999, 802.11b–1999, 802.11b–1999/Cor 1–2001, and 802.11d–2001].

[76] Paul Korzeniowski. Work on Higher-Speed WLAN Standard Begins. TechNewsWorld, 7th August 2004. ECT News Network, Inc. [Online] Available: `http://www.technewsworld.com/story/35607.html`

[77] Kelly Davis-Felner. Wi-Fi CERTIFIED 802.11n draft 2.0: Longer-Range, Faster-Throughput, Multimedia-Grade Wi-Fi Networks, 2007. Wi-Fi Alliance White Paper. [Online] Available: `http://www.wi-fi.org/files/kc/WFA_802_11n_Industry_June07.pdf`

[78] Eric Griffith. 802.11n: The Battle Begins. Wi-Fi Planet, 12th July 2004. Jupitermedia Corporation. [Online] Available: `http://www.wi-fiplanet.com/news/article.php/3379371`

[79] IEEE P802.11 Task Group N. Status of Project IEEE 802.11n. IEEE P802.11 — Task Group N — Meeting Update, November 2004. Latest update March 2009. [Online] Available: `http://www.ieee802.org/11/Reports/tgn_update.htm`

[80] Craig Mathias. My oh MIMO. Network World Fusion, 30th August 2004. Network World, Inc. [Online] Available: `http://www.nwfusion.com/research/2004/083004mimo.html`

[81] Mark Hachman. Industry Coalition Floats Proposal for 802.11n. eWeek, 12th August 2004. Ziff Davis Publishing Holdings, Inc. [Online] Available: `http://www.eweek.com/article2/0,1759,1635268,00.asp`

[82] Steven M. Cherry. Broader Broadband. *IEEE Spectrum*, 42(1):13–14, January 2005. International edition.

[83] Stephen McCann. Official ieee 802.11 working group project timelines, 16th March 2009. Institute of Electrical and Electronics Engineers, Inc. [Online] Available: `http://grouper.ieee.org/groups/802/11/Reports/802.11_Timelines.htm`

[84] John D. O'Sullivan, Graham R. Daniels, Terence M. P. Percival, Diethelm I. Ostry, and John F. Deane. *Wireless LAN*. Commonwealth Scientific and Industrial Research Organisation, Australia, us patent no. 5487069 edition, 23rd January 1996. (Application No. 157375 filed on 23 November 1993). [Online] Available: `http://www.patentstorm.us/patents/5487069/description.html`

[85] Asher Moses. CSIRO cashes in on patent claim. The Age newspaper, 1st April 2009. Fairfax Digital. [Online] Available: `http://www.theage.com.au/articles/2009/04/01/1238261630683.html`

[86] IEEE Std 802.11d–2001. *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 3: Specification for operation in additional regulatory domains*. IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 13th July 2001. (Amendment to IEEE Std 802.11, 1999 Edition, IEEE Std 802.11a–1999, and IEEE Std 802.11b–1999).

[87] IEEE Std 802.11F–2003. *IEEE Trial-Use Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation*. IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 14th July 2003.

[88] John Cox. The WLAN standards alphabet keeps growing. Network World Fusion, 13th December 2004. Network World, Inc. [Online] Available: `http://www.nwfusion.com/news/2004/121304specialfocus.html`

[89] Galen Gruman. Next challenges for wireless LANs. InfoWorld, 26th November 2004. International Data Group, Inc. [Online] Available: `http://www.infoworld.com/article/04/11/26/48FEwifispecs_1.html`

[90] Joanie Wexler. Mesh Moves Into the Wireless Office. Computerworld, 29th November 2004. International Data Group, Inc. [Online] Available: `http://www.computerworld.com/mobiletopics/mobile/technology/story/0,10801,97832,00.html`

[91] Chris Wullems and Kevin Tham and Jason Smith and Mark Looi. A Trivial Denial of Service Attack on IEEE 802.11 Direct Sequence Spread Spectrum Wireless LANs. In *Proc. Wireless Telecommunications Symposium, 2004*, pages 129 – 136. Information Security Research Centre, Queensland University of Technology, Brisbane, Australia, IEEE, May 2004.

[92] Changhua He and John C Mitchell. Security Analysis and Improvements for IEEE 802.11i. In *Network and Distributed System Security Symposium Conference Proceedings: 2005.* Electrical Engineering and Computer Science Departments, Stanford University, Stanford CA, The Internet Society, February 2005. [Online] Available: `http://www.isoc.org/isoc/conferences/ndss/05/proceedings/papers/NDSS05-1107.pdf`

[93] Joshua Wright. Detecting Wireless LAN MAC Address Spoofing, 21st January 2003. Johnson & Wales University. [Online] Available: `http://home.jwu.edu/jwright/papers/wlan-mac-spoof.pdf`

[94] Mike Martin. Researchers Study Wi-Fi Weaknesses. Enterprise Security Today, 25th October 2004. NewsFactor Network (article no longer available). [Online] Available: `http://enterprise-security-today.newsfactor.com/story.xhtml?story_id=27842`

[95] Jesse R. Walker. Unsafe at any key size; An analysis of the WEP encapsulation. Submission to the IEEE, 27th October 2000. Intel Corporation. [Online] Available: `http://www.dis.org/wl/pdf/unsafe.pdf`

[96] Anton T. Rager. WEPCrack — An 802.11 key breaker, 25th August 2001. SourceForge.net. [Online] Available: `http://wepcrack.sourceforge.net/`

[97] Snax. AirSnort Homepage, 15th September 2004. The Shmoo Group, airsnort-0.2.5. [Online] Available: `http://airsnort.shmoo.com/`

[98] Mike Kershaw. Kismet, 1st April 2004. [Online] Available: `http://www.kismetwireless.net/documentation.shtml`

[99] Christophe Devine. aircrack documentation, 4th January 2005. [Online] Available: http://www.cr0.net:8040/code/network/aircrack/

[100] Thomas d'Otreppe. Main [aircrack-ng]. DokuWiki, 4th August 2007. [Online] Available: http://www.aircrack-ng.org/doku.php

[101] José Ignacio Sánchez. Weplab, 11th September 2004. Update (mid 2005): The author now appears to be using the name José Ignacio Sánchez Martín. [Online] Available: http://weplab.sourceforge.net/

[102] Gerald Combs. Ethereal – The world's most popular network protocol analyzer, 13th May 2004. [Online] Available: http://www.ethereal.com/

[103] Marius Milner. NetStumbler v0.4.0 Release Notes, 21st April 2004. [Online] Available: http://www.stumbler.net/readme/readme_0_4_0.html

[104] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In *Proc. Eighth Annual Workshop on Selected Areas in Cryptography*. Springer-Verlag, August 2001.

[105] The Shmoo Group. Airsnarf – A rogue AP setup utility, 4th November 2003. [Online] Available: http://airsnarf.shmoo.com/

[106] Andrew Brandt. A latte, a Wi-Fi link and a hacker. PC World, 25th November 2003. International Data Group, Inc. [Online] Available: http://www.computerworld.com/securitytopics/security/story/0,10801,87523,00.html

[107] Michael Lynn and Robert Baird. AirJack, 14th February 2004. (no longer available). [Online] Available: http://802.11ninja.net/airjack/

[108] Michael Ossmann. Wep: Dead again. SecurityFocus Infocus, 14th December 2004. Part 1 published Dec 2004, Part 2 published Mar 2005. [Online] Available: http://www.securityfocus.com/infocus/1814

[109] David Hulton. dwepcrack v0.4, January 2002. dachb0den labs: projects: bsd-airtools: dweputils: dwepcrack. [Online] Available: http://www.dachb0den.com/projects/dwepcrack.html

[110] Stephen Kent and Karen Seo. *Security Architecture for the Internet Protocol.* BBN Technologies, December 2005. IETF Network Working Group, RFC 4301, Standards Track, Obsoletes RFC 2401.

[111] IEEE Std 802.1X–2001. *IEEE Standard for Local and metropolitan area networks—Port-Based Network Access Control.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 13th July 2001.

[112] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL Protocol Version 3.0. Transport Layer Security Working Group, Internet-Draft, 18th November 1996. Netscape Communications, draft-freier-ssl-version3-02.txt. [Online] Available: `http://wp.netscape.com/eng/ssl3/draft302.txt`

[113] Kipp E.B. Hickman. The SSL Protocol. Draft RFC submitted to the W3O working group on security, 29th November 1994. Netscape Communications Corp. [Online] Available: `http://www.llnl.gov/atp/papers/HRM/references/ssl.html`

[114] Kipp E.B. Hickman. The SSL Protocol. updated draft RFC, Category: Informational, 9th February 1995. Netscape Communications Corp. [Online] Available: `http://wp.netscape.com/eng/security/SSL_2.html`

[115] Michael Hayoz. Introducing SSL — The Secure Sockets Layer Protocol. MSc Seminar in Telecommunications, 16th June 2003. Department of Informatics, University of Freiburg i. Ue., Switzerland. [Online] Available: `http://diuf.unifr.ch/ds/michael.hayoz/docs/hayozm_ssl.pdf`

[116] Cisco Systems, Inc. Cisco Aironet Response to University of Marylands Paper, "An Initial Security Analysis of the IEEE 802.1x Standard" [118], 2002. [Online] Available: `http://www.cisco.com/warp/public/cc/pd/witc/ao350ap/prodlit/1680_pp.pdf`

[117] Cisco Systems, Inc. User Guide for Cisco Secure ACS for Windows 4.0, 26th March 2006. [Online] Available: `http://www.cisco.com/en/US/docs/net_mgmt/cisco_secure_access_control_server_for_windows/4.0/user/guide/user.html`

[118] Arunesh Mishra and William A. Arbaugh. An Initial Security Analysis of the IEEE 802.1x Standard, 6th February 2002. Department Of Computer

Science, University Of Maryland. [Online] Available: `http://www.cs.umd.edu/~waa/1x.pdf`

[119] IEEE Std 802.11h–2003. *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 5: Spectrum and Transmit Power Management Extensions in the 5 GHz band in Europe.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 14th October 2003. (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003), as amended by IEEE Stds 802.11a–1999, 802.11b–1999, 802.11b–1999/Cor 1–2001, 802.11d–2001, and 802.11g–2003).

[120] IEEE Std 802.11j–2004. *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 7: 4.9 GHz-5 GHz Operation in Japan.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 29th October 2004. [Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003) as amended by IEEE Stds 802.11a–1999, 802.11b–1999, 802.11b–1999/Cor 1–2001, 802.11d–2001, 802.11g–2003, 802.11h–2003, and 802.11i–2004].

[121] IEEE Std 802.11e–2005. *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 11th November 2005. (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003) as amended by IEEE Stds 802.11a–1999, 802.11b–1999, 802.11b–1999/Cor 1–2001, 802.11d–2001, 802.11g–2003, 802.11h–2003, 802.11i–2004, and 802.11j–2004).

[122] IEEE Std 802.11–2007 (Revision of IEEE Std 802.11–1999). *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific*

requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 12th June 2007. Approved 8 March 2007.

[123] IEEE Std 802.11k–2008. *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Radio Resource Measurement of Wireless LANs.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 12th June 2008. (Amendment to IEEE Std 802.11–2007).

[124] IEEE Std 802.11r–2008. *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 2: Fast Basic Service Set (BSS) Transition.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 15th July 2008. (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008).

[125] Dan Simone. 802.11k makes WLANs measure up, 29th March 2004. Network World, Inc. [Online] Available: `http://www.networkworld.com/news/tech/2004/0329techupdate.html`

[126] Adam Stubblefield, John Ioannidis, and Aviel D. Rubin. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP. Technical Report TD-4ZCPZZ, AT&T Labs, 6th August 2001.

[127] David Halasz. IEEE 802.11i and wireless security. Embedded.com, 25th August 2004. CMP Media LLC. [Online] Available: `http://www.embedded.com/showArticle.jhtml?articleID=34400002`

[128] Defence Signals Directorate (Australian Department of Defence). *Australian Government Information and Communications Technology Security Manual (ACSI 33)*, 1st March 2005.

[129] Dorothy Stanley. TGu/TGv Ad-hoc Meeting. Submission, 30th April 2008. [Online] Available: `https://mentor.ieee.org/802.11/file/08/11-08-0491-00-000v-tgu-tgv-ad-hoc-april-2008.ppt`

[130] Joseph Davies. Wi-Fi Protected Access 2 (WPA2) Overview. Microsoft TechNet — The Cable Guy, 15th June 2005. Microsoft Corporation. [Online] Available: `http://www.microsoft.com/technet/community/columns/cableguy/cg0505.mspx`

[131] Kwang-Hyun Baek, Sean W. Smith, and David Kotz. A Survey of WPA and 802.11i RSN Authentication Protocols, November 2004. [Online] Available: `http://www.cs.dartmouth.edu/\~{}dfk/papers/baek:survey-tr.pdf`

[132] Gary Hughes. The cyberspace invaders. The Age, Melbourne, Australia, 22nd June 2003. The Age Company Ltd, f2 network, Fairfax interactive network. [Online] Available: `http://www.theage.com.au/articles/2003/06/21/1056119529509.html`

[133] John Cox. DARPA looks to adaptive battlefield wireless nets. Network World, 1st November 2007. Network World, Inc. [Online] Available: `http://www.networkworld.com/news/2007/110107-wand.html`

[134] Defense Advanced Research Projects Agency (DARPA). Wireless Network after Next Program, 29th September 2005. [Online] Available: `http://www.darpa.mil/sto/solicitations/WNaN/index.htm`

[135] John Sherwood, Andrew Clark, and David Lynas. *Enterprise Security Architecture: A Business-Driven Approach.* CMP Books, 2005.

[136] Jeff Tendero, Vanessa Freke, Allan Klason, and Jason Jones. Queensland Government Authentication Framework. Queensland Government Chief Information Office, October 2006. Endorsed Version 1.0.1, following PUBLIC classification.

[137] Luke Turner and David Ross. 802.1X — from business drivers to implementation. Bridge Point Communications ‘TechFast’ (vendor technical breakfast briefing), February 2009.

[138] D. Simon, B. Aboba, and R. Hurst. *The EAP-TLS Authentication Protocol.* Microsoft Corporation, March 2008. IETF Network Working Group, RFC 5216, Standards Track, Obsoletes RFC 2716.

[139] P. Funk and S. Blake-Wilson. *Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0).* Funk Software and SafeNet, August 2008. IETF Network Working Group, RFC 5281.

[140] N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou. *The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST).* Cisco Systems, May 2007. IETF Network Working Group, RFC 4851.

[141] Carol Ellison. Wi-Fi Alliance Expands Enterprise Product Certification. eWeek, 12th April 2005. Ziff Davis Enterprise Holdings Inc. [Online] Available: `http://www.eweek.com/c/a/Mobile-and-Wireless/WiFi-Alliance-Expands-Enterprise-Product-Certification/`

[142] Wi-Fi Alliance. Wi-Fi CERTIFIED expanded to support EAP-AKA and EAP-FAST authentication mechanisms. press release, 19th May 2009. [Online] Available: `http://www.wi-fi.org/pressroom_overview.php?newsid=817`

[143] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, Ed. *Extensible Authentication Protocol (EAP).* Microsoft, Merit Network, Vollbrecht Consulting, Sun Microsystems and ipUnplugged, June 2004. IETF Network Working Group, RFC 3748, Standards Track, Obsoletes RFC 2284.

[144] Donald Welch and Scott Lathrop. Wireless Security Threat Taxonomy. In *Proceedings of the 2003 IEEE Workshop on Information Assurance, 2003*, pages 76–83, United States Military Academy, West Point, NY, 18–20thJune 2003. IEEE Systems, Man and Cybernetics Society.

[145] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. Breaking 104 bit WEP in less than 60 seconds. Cryptology ePrint Archive, Report 2007/120, 1stApril 2007. last revised 16 Sep 2007. [Online] Available: `http://eprint.iacr.org/2007/120`

[146] Joshua Wright. 802.11w security won't block DoS attacks. Network World, accessed via Techworld, 14th June 2006. International Data Group, Inc. [Online] Available: `http://www.techworld.com/mobility/features/index.cfm?featureid=2599`

[147] IEEE Standards Board Project Authorization Request (PAR) P802.11w. *Standard for Information Technology—Telecommunications and Information Exchange between systems—Local and Metropolitan networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Protected Management Frames Amendment.* IEEE Standards Association, 20th March 2005. (Amendment to an Existing IEEE Standard 802.11–1999).

[148] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 162–175. ACM, November 2004. ISBN: 1-58113-879-2.

[149] Chris Hurley. The WorldWide WarDrive 4. In Jeff Moss, editor, *Proc. Black Hat USA Briefings 2004*, pages 561–568, Las Vegas, July 2004. Black Hat, Inc. Slides published. Context sourced direct from the conference presentation.

[150] Matthew Gast. *802.11 Wireless Networks: The Definitive Guide.* O'Reilly and Associates, Inc., April 2002.

[151] Fortress Technologies. Fortress technologies. Company History public webpage, 2007. [Online] Available: `http://www.fortresstech.com/about_us/history.asp`

[152] Raj Pandya. *Introduction to WLLs: Application and Deployment for Fixed and Broadband Services.* John Wiley & Sons — IEEE Press, December 2003. Book precis available on publisher's site. [Online] Available: `http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471451320,miniSiteCd-IEEE2.html`

[153] Joshua Robinson. Making NS-2 simulate an 802.11b link, 29th April 2005. [Online] Available: `http://www.ece.rice.edu/~jpr/ns/docs/ns-802_11b.html`

[154] Steven McCanne and Sally Floyd. ns Network Simulator, 1995. [Online] Available: `http://www.isi.edu/nsnam/ns/`

[155] David Harrison, Paul Walker, and the VINT project. xgraph, 1999. The VINT release of David Harrison's xgraph. [Online] Available: `http://www.isi.edu/nsnam/xgraph/index.html`

[156] Thomas Williams and Colin Kelley. gnuplot, 1986. [Online] Available: `http://www.gnuplot.info/`

[157] Srinivasan Keshav. REAL : A Network Simulator. Technical report, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley., December 1988.

[158] Alexander Dupuy, Jed Schwartz, Yechiam Yemini, and David Bacon. NEST: a network simulation and prototyping testbed. *Communications of the ACM*, 33(10):63–74, October 1990. Special issue on simulation, ISSN:0001-0782.

[159] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. Developing the Next-Generation Open-Source Network Simulator (ns-3). CRI Proposal #0551686, 24th June 2006. University of Washington, ICSI Center for Internet Research and Georgia Institute of Technology.

[160] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. ns-3 project goals. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator (WNS2)*, Pisa, Italy, 2006. ACM, New York, NY, USA.

[161] Dennis M. Ritchie and Ken Thompson. The UNIX Time-Sharing System. *Communications of the ACM*, 17(7):365–375, July 1974. Bell Laboratories.

[162] Sally Floyd. Simulator Tests. Technical report, Lawrence Berkeley Laboratory, One Cyclotron Road, Berkeley, CA 94704, May 1997. [Online] Available: `ftp://ftp.ee.lbl.gov/papers/simtests.ps.Z`

[163] Kevin Fall and Kannan Varadhan, editors. *The ns Manual (formerly ns Notes and Documentation)*. The VINT project (UC Berkeley, LBL, USC/ISI, and Xerox PARC), 14th April 2002. [Online] Available: `www.isi.edu/nsnam/ns/doc/ns_doc.pdf`

[164] Ke Liu. Understanding the implementation of ieee mac 802.11 standard in ns-2, 3rd May 2006. [Online] Available: `http://www.cs.binghamton.edu/~kliu/research/ns2code/note.pdf`

[165] Ilango Purushothaman and Sumit Roy. Technical Report — IEEE 802.11 implementation issues/bugs in ns2. Technical report, Department of EE, University of Washington, Seattle, WA, USA, March 2007.

[166] Ilango Purushothaman and Sumit Roy. Infrastructure mode support for IEEE 802.11 implementation in NS-2. Technical report, Department of EE, University of Washington, Seattle, WA, USA, December 2007.

[167] Thomas R. Henderson. in ns: Change History. ns-2.34 pending, 13th December 2008. Bug fixes for 802.11 infrastructure code, posted by Ilango Purushothaman. [Online] Available: `http://www.isi.edu/nsnam/ns/CHANGES.html`

[168] Martina Umlauft and Peter Reichl. Experiences with the ns-2 Network Simulator — Explicitly Setting Seeds Considered Harmful, March 2007. Vienna University of Technology and Telecommunications Research Center Vienna (FTW). [Online] Available: `http://publik.tuwien.ac.at/files/pub-inf_4620.pdf`

[169] rand(3). *rand, rand_r, srand — pseudo-random number generator.* The Linux Kernel Organization, Inc., 29th August 2008. Release 3.16 of the Linux man-pages project. [Online] Available: `http://www.kernel.org/doc/man-pages/`

[170] Pierre Ferré, Angela Doufexi, Andrew Nix, and David Bull. Throughput Analysis of IEEE 802.11 and IEEE 802.11e MAC. In *Proc. IEEE Wireless Communications and Networking Conference, 2004 (WCNC)*, volume 2, pages 783–788, 21–25 March 2004.

[171] Nakjung Choi, Yongho Seok, Yanghee Choi, Sungmann Kim, and Hanwook Jung. P-DCF: Enhanced Backoff Scheme for the IEEE 802.11 DCF. In *Proc. IEEE 61st Vehicular Technology Conference, 2005 (VTC 2005-Spring)*, volume 3, pages 2067–2070, 30 May–1 June 2005.

[172] Huan-Chun Lin, Ya-Ching Chang, I-Chien Liu, and Shih-Tsung Liang. Study on the Enhanced Back-off Schemes for the IEEE 802.11 Wireless LANs. In *Proc. 4th IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS 2007)*, National Chiao Tung University, Hsinchu, Taiwan, 20–21 August 2007. Dept. Math. Computer Science Education, Taipei Municipal University of Education, No.1, Ai-Guo West Road,Taipei,10042, Taiwan.

[173] Jun Lv, Xinming Zhang, Xiaojun Han, and Yanyan Fu. A Novel Adaptively Dynamic Tuning of the Contention Window (CW) for Distributed Coordination Function in IEEE 802.11 Ad hoc Networks. In *Proc. The 2007 International Conference on Convergence Information Technology*, pages 290–294, Gyeongju, 21–23 November 2007.

[174] ANSI/IEEE Std 802.11, 1999 edition (R2003). *Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access control (MAC) and Physical Layer (PHY) Specifications.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 20th August 1999. Approved 18 March 1999. Reaffirmed 12 June 2003.

[175] ISO/IEC 8802-11: 2005(E); IEEE Std 802.11, 2003 Edition. *Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.* IEEE Standards Association, 3 Park Avenue, New York, NY, USA, 1st August 2005.

[176] Mohammad Ilyas and Imad Mahgoub. *Smart Dust: Sensor Network Applications, Architecture, and Design.* CRC Press, 2006.

[177] Giuseppe Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, March 2000.

[178] Haitao Wu, Shiduan Cheng, Yong Peng, Keping Long, and Jian Ma. IEEE 802.11 Distributed Coordination Function(DCF): Analysis and Enhancement. In *Proc. IEEE International Conference on Communications, 2002 (ICC 2002)*, volume 1, pages 605–609, 2002.

[179] Qiang Ni, Imad Aad, Chadi Barakat, and Thierry Turletti. Modeling and analysis of slow CW decrease IEEE 802.11 WLAN. In *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, 2003 (PIMRC 2003)*, volume 2, pages 1717–1721, 7th–10th September 2003.

[180] Nai-Oak Song, ByungJae Kwak, Jabin Song, and Leonard E. Miller. Enhancement of IEEE 802.11 distributed coordination function with exponential increase exponential decrease backoff algorithm. In *Proc. The 57th IEEE Semiannual Vehicular Technology Conference, 2003 (VTC 2003-Spring)*, volume 4, pages 2775–2778, 22–25 April 2003.

[181] Nicola Baldo. Developer's manual. part of the Dynamic Modules in NS: patch documentation, 9th November 2007. University of Padova — Department of Information Engineering. [Online] Available: `http://news.com.com/2100-7351_3-5182185.html`

[182] C. Sivakumar and A .Velmurugan. High Speed VLSI Design CCMP AES Cipher for WLAN (IEEE 802.11i). In *Proc. International Conference on Signal Processing, Communications and Networking, 2007 (ICSCN '07)*, pages 398–403, 22–24 February 2007. ISBN: 1-4244-0997-7.

[183] Paul W. Frields. Infrastructure report, 2008-08-22 UTC 1200. e-mail To: fedora-announce-list ¡fedora-announce-list@redhat.com¿, 22nd August 2008. [Online] Available: `http://www.redhat.com/archives/fedora-announce-list/2008-August/msg00012.html`