THE LAST WALTZ AND MOVING BEYOND TCP/IP

John Nolan

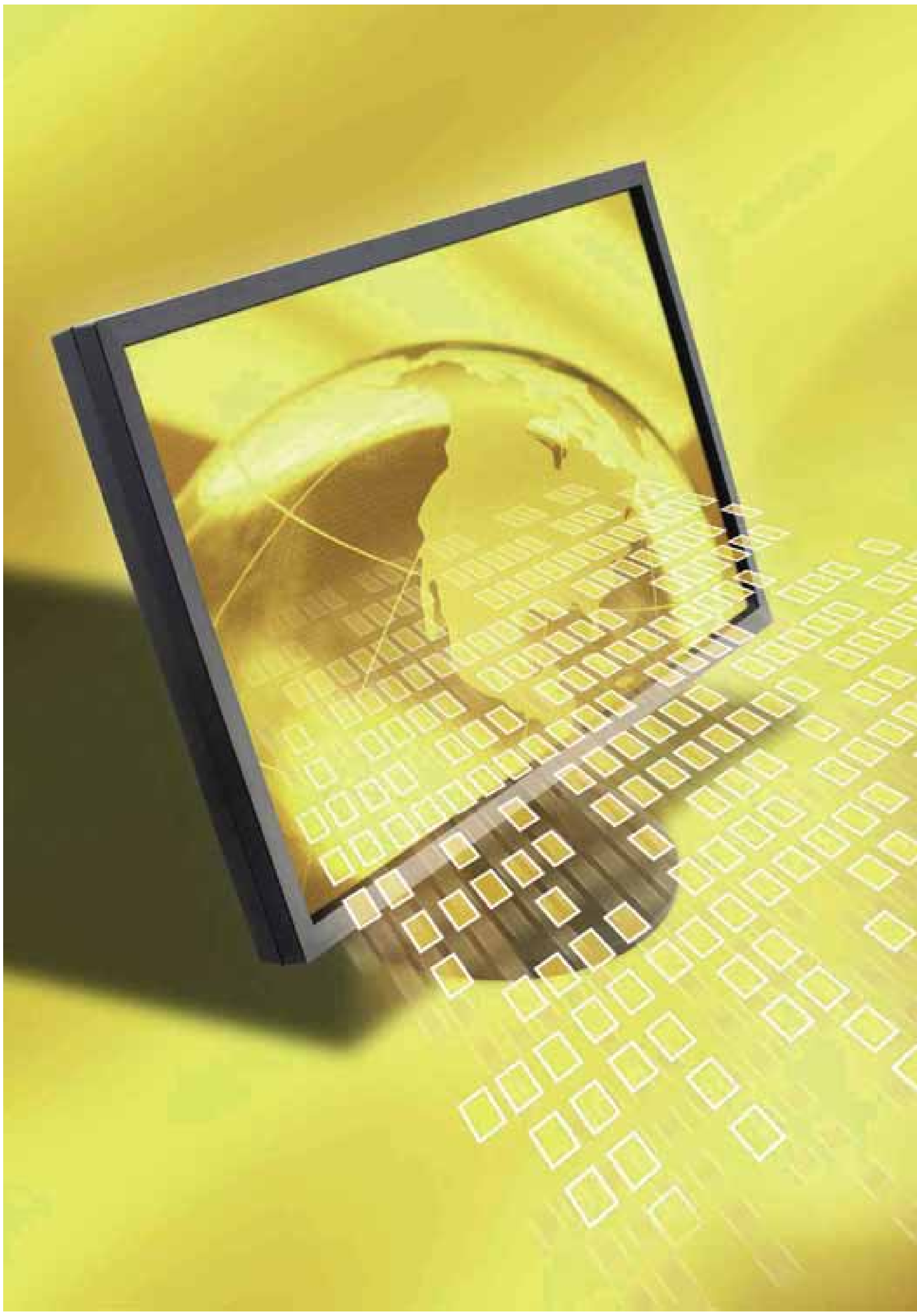
Is the TCP/IP protocol suite approaching its Last Waltz? John Nolan poses the question after examining the work of John Day, following a review of his book, Patterns in Network Architecture: A Return to Fundamentals within the IP Journal [1]. Further research then led him to The Pouzin Society [2] and in particular to the paper that follows on these pages[1].

Moving beyond TCP/IP[1]

Fred Goldstein and John Day for the Pouzin Society - June 2011.

The triumph of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite in today's market is nearly complete. A monoculture of networking has emerged, based on protocols originally developed in the 1970s. With a near-universal use of IP for purposes well beyond the original designers' intent, conventional wisdom holds that all future solutions must slowly evolve from it.

This belief, however popular, is not necessarily correct. The Internet itself has been a popular success in large part because of its low-price business model. IP has absorbed the glow from the Internet's halo. People confuse the Internet with its protocols. But they are not the same thing. TCP/IP has been a 30-year distraction from real internetworking. In a real sense, it is the networking equivalent of Microsoft DOS (Disk Operating System). For the Internet to prosper in the long term, it needs to move beyond TCP/IP.



TCP/IP WAS DESIGNED FOR A LIMITED SET OF TASKS

TCP/IP was designed for the Advanced Research Projects Agency Network (ARPANET), a Department of Defense resource-sharing network. When the ARPANET began in 1969, it demonstrated the then-radical notion of packet switching. The original ARPANET protocol, Network Control Program (NCP), was designed to ensure reliability of transmission on a hop-by-hop basis. That ARPANET is not today's Internet. It was more like the X.25 packet-switched networks that were developed later in the 1970s.

A French researcher, Louis Pouzin, postulated that the switches in the middle of the network didn't have to keep track of connections; they just had to pass packets as they arrived. Error correction and flow control could be handled at the edges of the network. He designed and built the first connectionless network, CYCLADES, in 1972. But there was much work still to do when political pressure shut down the project.

ARPANET developers adopted the connectionless idea, but failed to see the work that was still needed to create a basic architecture They created a new set of protocols, with TCP for end-to-end error and flow control and IP (which was separated from TCP in Version 4) for the connectionless middle. It was an "internet" because it originally ran atop other networks, such as NCP and X.25, allowing their respective users to share information. Certain questions, like congestion control, still needed to be answered, but as key personnel changed in the late 1970s, they were forgotten.

With 1983's "flag day", TCP/IP had completely replaced NCP, and IP, began to see widespread use as a network protocol. Around that time, Berkeley released a free, open source Berkeley Software Distribution Unix implementation of TCP/IP, including key applications. While rather crude, the price was right, subsidised by US tax dollars. And it provided for vendor-independent networking just as International Standards Organisation's Open Systems Interconnection (OSI) standards project was tearing itself apart from internet conflicts.

> "The need to grossly over-provision to avoid congestion in the Internet backbone allowed streams to work just well enough to catch on."

TCP/IP's strength was it worked for the current environment and was free. Moore's Law allowed it to keep up with growth and covered up other holes. It easily handled the current applications, such as file and print services, although the World Wide Web provided a few bumps. But packet switching was designed to handle bursty data. IP was not optimised to support streaming.

IP has absorbed the glow from the Internet's halo – but TCP/IP has been a 30 year distraction from real internetworking.

Disclaimer: Neither John Nolan or First Mile Networks have any association whatsoever with the Pouzin Society, apart from an academic interest.

The TCP/IP ARPANET lacked any kind of security mechanisms, depending entirely on the security of the computers connected to it.

The need to grossly over-provision to avoid congestion in the Internet backbone allowed streams to work just well enough to catch on. Good enough was the enemy of product quality. The IP juggernaut was unstoppable.

IP HAS A NUMBER OF WEAKNESSES TODAY

The TCP/IP ARPANET itself had weaknesses that are surprising in a network financed by national security dollars. It lacked any kind of security mechanisms, depending entirely on the security of the computers connected to it. (We see today how well that worked out!) And it lacked support for redundant connections (multihoming). By the 1980s, TCP/IP was the most open protocol stack but not the most powerful.

And the Internet of today isn't the ARPANET of the 1970s or for that matter the Internet of the early 1990s. There are many issues that IP doesn't handle well and which could be addressed if a new protocol were developed from a clean slate, rather than as an extension of IP. TCP/IP was a research project; let's exploit the results of that research.

IP LACKS A COMPLETE ADDRESSING ARCHITECTURE

IP's addressing architecture is incomplete. An IP address refers to a point of attachment (POA), not a node, and a route is thus a series of POAs. This makes multihoming almost useless, since each connection has a different address. Routing should be to the node, not the POA. This mistake was perpetuated by the IETF in IP Version 6. The problem is magnified when dealing with multihomed networks. Essentially, every backbone router needs to keep track of links to every network. Hundreds of thousands of them. Interconnection between networks thus remains very sparse as it was in the 1980s. Network designers have to strike a balance between creating too many links, and thus overburdening the routers, and having too few links, and thus having to send local traffic on a very roundabout route.

This doesn't scale well at all. The more networks on the Internet, and the more multihomed provider-independent address blocks, the more routes there are, and the number of possible paths thus rises faster than linearly. Oops. Add new demand from things, like "Smart Grid" that needs to multihome every meter, and it gets far worse.

How does IPv6 handle this? By having a larger address space, it permits even more address blocks to exist. It doesn't change the architecture; it just speeds up the fuel pump feeding the fire. IPv6 was designed to maintain the status quo.

NAT IS YOUR FRIEND

Network Address Translation (NAT) is a controversial part of the TCP/IP world. It serves two major functions. It conserves IP addresses, and it adds a layer of security. NAT is often seen as a layer violation because the NAT device has to modify the TCP and IP layers together, changing port numbers (in TCP or User Datagram Protocol) as well as IP addresses. This is not a problem with NAT per se. IP address + port number is the connection identifier. The two protocols should be viewed as being in same layer. So address translation naturally deals with them together. NATs only break broken architectures.

There is, of course, one clear layer violation that NAT has to deal with, but that's not NAT's fault either. Some application protocols put an IP address inside the application layer header. These have to be modified by NAT, so a NAT has to understand their syntax. This is rather like passing physical memory addresses in a Java program.

Another problem with TCP/IP is that the real name of an application is not the text form that humans type; it's an IP address and its well-known-port number. As if an application name were a macro for a jump point through a well-known low memory address. When an application uses a host name or other text form such as a URI, the application resolves it by querying a Domain Name System (DNS). Resolving names in the network layer would provide a cleaner mechanism for applications to find each other.

In fact, even host names should not be used in the application. Applications, not hosts, should be what's named. The host simply hosts it. Many applications nowadays run on more than one host. A popular application might run on thousands of hosts distributed worldwide (think Google). This requires a royal kludge in the TCP/IP architecture

IP IS POORLY SUITED FOR STREAMING

IP was designed to deliver packets on a "best efforts" basis, meaning that it's okay to throw packets away. That's not a bad thing. But it works best when the payload can use retransmission. Streaming means that there's no time to retransmit, so the stream must have low loss. The best that IP can do for streaming is to assign priorities, with high priority given to streams, such as telephone calls.

When a packet is lost, ordinary data using TCP slows down; streams do not. So if there are too many streams on a link, ordinary data can be crowded out. We already have a telephone network and cable TV; these applications could potentially break the unique capabilities that only the Internet can provide. The telephone industry is evolving towards the use of IP, but it is a tricky proposition. Telephone streams often have to be separated at a lower layer, put into separate flows using MPLS, Carrier Ethernet, or some other technique, even Time Division Multiplex. These offer the lossless assurance of bandwidth that IP, being connectionless, lacks. Attempts to handle streaming within IP are preposterously complex and unproven. (See, for instance, IMS, the IP Multimedia Subsystem. It's the nuclear fusion of IP: It's always a few years from being ready.)

A PATH FORWARD

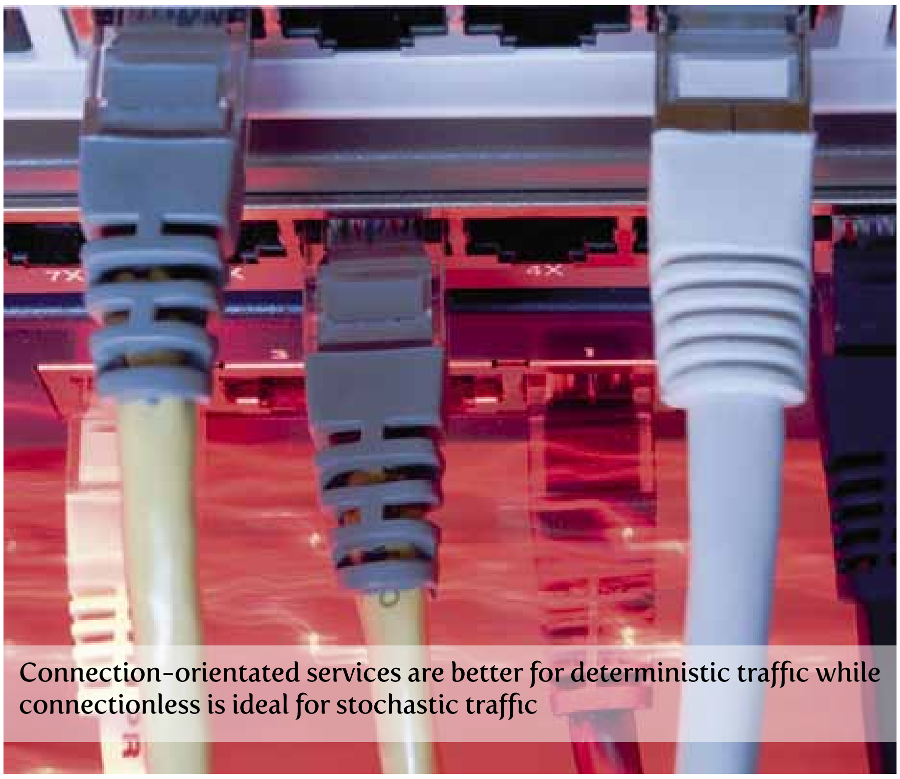
So if IP is so imperfect, can anything be done about it? Of course… but it's not going to be handled by incremental upgrades or missteps like IPv6. Instead, what John Day has done in his Patterns in Network Architecture: A Return to Fundamentals [1] is start afresh, taking into account lessons learned in the 35 years of TCP/IP's existence, as well as the lessons of the 1980s' failed OSI program, and the lessons of other network technologies of the past few decades. He has made some key observations that point to a new direction.

OSI's famous 7-layer reference model was defined too early, and incorrectly separated out Layers 5, 6 and 7, which should have been one layer, while there were potentially 7 layers below the application. But fixed protocol stacks themselves turn out to be the problem! The pattern that Day noted is that protocol functions are repeated in different layers,

"So if IP is so imperfect, can anything be done about it? Of course ... but it's not going to be handled by incremental upgrades or missteps like IPv6."

which differ in policy and scope. This alternation reflects a repeating unit consistent with interprocess communication. The error and flow control function breaks up into two functions, data transfer, which transfers data, and data transfer control, which provides feedback from the receiver (ack and flow control).

The defining characteristic of a layer in networks is distributed shared state of a given scope. More than one protocol may exist within a layer, but layers should be opaque to one another. All layers do the same thing for a different range of the problem. By separating policy from mechanism, the same protocol mechanism can work across wide


Connection-orientated services are better for deterministic traffic while connectionless is ideal for stochastic traffic

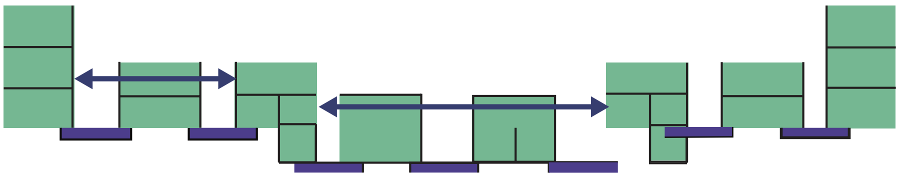
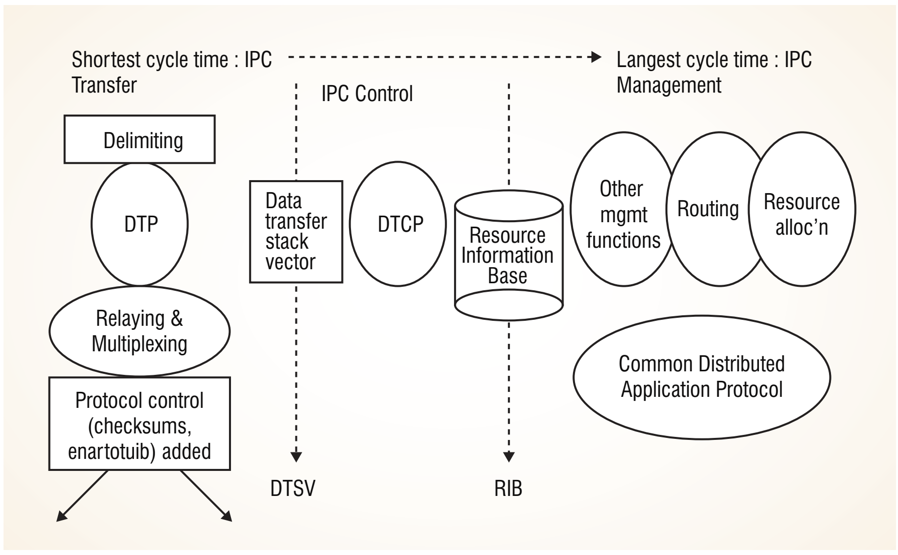


Figure 1: *DIFs may be stacked as required. The number of layers in the stack may vary from hop to hop*



Shortest cycle time : IPC Transfer
Langest cycle time : IPC Management
IPC Control
Delimiting
DTP
Relaying & Multiplexing
Protocol control (checksums, enartotuib) added
Data transfer stack vector
DTCP
Resource Information Base
Other mgmt functions
Routing
Resource alloc'n
Common Distributed Application Protocol
DTSV
RIB

Figure 2: *Functions are executed within a DIF over different time scales*

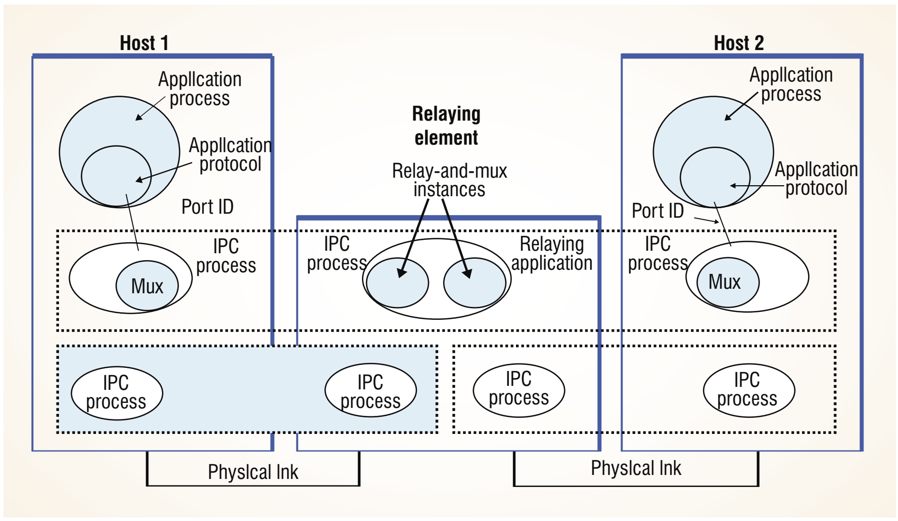


Figure 3: *A router is a DIF which performs relaying between DIFs as its application.*

ranges of scope, QoS, and bandwidth. Lower layers thus to do the same things as upper layers. Near the bottom, their characteristics are dominated by the media; further up it is the applications that dominate.

RECURSIVE LAYERS

Networking is just interprocess communications (IPC); ultimately all networking is communication between processes. IPC within a single computer is quite simple; it could just take the form of memory shared between two processes. IPC between computers requires additional mechanisms to deal with issues such as reliability of communications and synchronisation. That's where network protocols come in.

This leads to the first principle of our proposed new network architecture: Layers are recursive. The same protocol can be used repeatedly in a protocol stack. There is thus no need for purpose-built protocols for each layer. There is not a fixed number of layers in the stack. The number of layers in any given network is variable. There are simply as many as needed, no more, no less. But any given system only has as many layers as it has in a current system today (see Figure 1). The actual depth of the stack is essentially invisible.

Because the same group of protocols is used repeatedly, the implementation is simpler than the TCP/IP stack. Because the layers recurse, and can scale to form a large internet, the protocol suite that supports the concept from Patterns in Network Architecture is called the Recursive Internetwork Architecture (RINA).

A RINA layer contains two protocols, the Error and Flow Control Protocol (EFCP) and the Common Distributed Application Protocol (CDAP). These are, in fact, the only two protocols in RINA and Figure 2 shows these protocols and the associated functions. EFCP includes subparts, the first being the Data Transfer Protocol (DTP) which contains addressing, fragmentation and protection information (checksum and time-to-live counter), and the second being the Data Transfer Control Protocol (DTCP) which sends feedback from destination to source. Note that the DTP's payload exists outside of the layer, while the DTCP operates entirely within that black box.

Routing (relaying) is just an application (see Figure 3), and the basic layer mechanism is called a Distributed IPC Facility, or DIF. It is a black box that operates among multiple systems; the IPC process thus runs on every system that belongs to the DIF. A DIF enforces strict layer boundaries: what happens inside the DIF is not visible outside of the DIF; what is visible at the top of a DIF is the service requested of the DIF; what is visible at the bottom is the service requested by the DIF of the DIF beneath it, if it isn't the bottom one.

Each DIF identifies the DIF or application above it by name. A DIF can span many systems; it may rely on the services of lower-layer DIFs to link the members of the DIF. And these DIFs may in turn rely on DIFs beneath them, transparent to the application processes that use them.

One of the capabilities of the DIF is the optional encryption. The DIF itself is a securable container; thus the need for firewalls has been eliminated. Service Data Unit protection is available at the bottom of every layer. Hence there is no need for a separate RINA equivalent of IPsec (Internet Protocol Security) or SSL (Secure Socket Layer). Deep packet inspection is thus impossible and irrelevant. Network content is thus inherently neutral, though not in the "one size fits all" sense desired by many IP neutrality advocates.

EFCP is based on Richard Watson's 1978 work that proves the necessary and sufficient condition for reliable transfer is to bound three timers. It does not require a connection set-up or tear-down for integrity purposes, like TCP's SYN and FIN. In fact, TCP uses the same three timers! So RINA lacks unnecessary overhead. Synchronisation and port allocation are distinct functions. This improves and simplifies security.

RINA benefits include improved scalability and security – in terms of both privacy of communications and protection against attack

Just as there are patterns visible in the other layers, it has become clear that only six fundamental operations can be performed remotely: Create/delete, read/write, and start/stop to objects external to the protocol. Hence it is clear that only one application protocol is required. RINA has adapted a previously existing protocol (CMIP - Common Management Information Protocol) and renamed it CDAP, or Common Distributed Application Protocol.

What changes from application to application are in the objects being manipulated, not the protocol. The applications themselves are outside of the protocol, while data transfer takes place within it. So the protocol remains stable, while new applications are accommodated easily by changing the object models. CDAP is also used within the DIF for layer management functions, including enrollment (joining the DIF) and security management, port allocation, access control, QoS monitoring, flow management, and routing.

DESIGNED FOR MULTIPLE FUNCTIONS

TCP/IP was designed for the data networking functions of the day, and was not intended to support VoIP and IPTV which rely on substantial over-provisioning. The work that led to RINA began by asking the question, why do all of these protocols look so much alike?

Investigation showed that there were a small number of mechanisms in a protocol that had policies. For example, acknowledgement is a mechanism, but when to ask is policy. Recognising that differences in syntax were minimal, it became clear that ultimately there was only one data transfer protocol. The entire range could be accommodated by different policies.

A DIF can be asked to provide a low-loss connection for a specified level of capacity, such as might be required for an audio or video stream. Or it can be asked for a best-effort (unspecified QoS) connection. RINA simply considers QoS to be a set of parameters plugged in to the DIF, and the specific mechanisms that it uses to provide it are an implementation detail, hidden inside the black box. If the DIF determines that it can't deliver the requested QoS, it rejects the request.

INTEGRATING CONNECTION-LESS AND CONNECTION-ORIENTED NETWORKING

One of the reasons that many TCP/IP backers are so passionate about their cause is that they fought hard battles over it. Connectionless service is fine for many things; the TCP/IP Internet is an existence proof that it works, something that the backers of the OSI connection-oriented network service (CONS), based on X.25, would not have believed possible. But as it turns out, the distinction between the two is less than it seems. The usual argument for connectionless networking is simplicity. By not promising too much, the network is

"Mobility is obviously nothing more than dynamic multi-homing, i.e. changing points of attachment more often than usual."

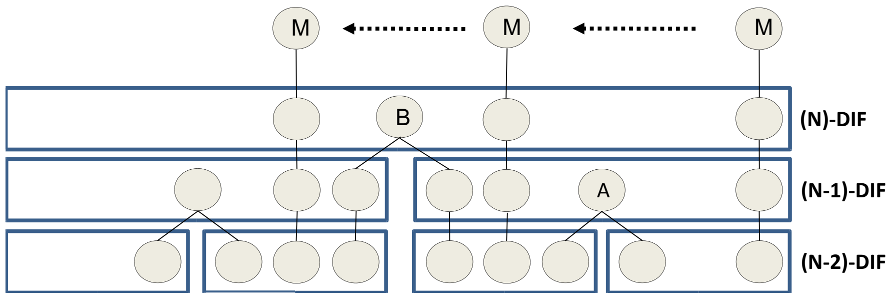


Figure 4: *As a mobile host moves, it joins new DIF and drops its participation in old ones*

simpler, or "stupid". A protocol has been considered connection-oriented if it needs to set up a context for the flow of data. IP is considered connectionless because each packet is just routed as it comes along.

But wait: How does a connectionless (IP) router know where to route packets to? We've already determined that an IP address doesn't actually tell you where it's going; it just names the destination network and interface. So in fact connectionless routers require more context than connection-oriented ones because they need to maintain, in effect, connections to every other node on the Internet! That's what routing tables are all about, and why they're getting bigger by the day.

Connectionless services are more robust to failures, while connection-oriented services are more brittle to failure. Each is good in its proper place: Connectionless is ideal for stochastic traffic; connections are better for a deterministic traffic. RINA achieves a unification of the connection/connectionless models. This sometimes is manifested as connectionless within a subnet but connections over the subnet.

NAMES ARE GLOBAL, ADDRESSES LOCAL

Addressing in the Internet defines a path from the interface to the application (IP address plus well-known port). DNS is merely a human-friendly macro for the IP address. One of the newest major protocols in the TCP/IP suite, HTTP - Hypertext Transfer Protocol (invented in 1989), tries to improve upon this; the actual URL (Uniform Resource Locater) string, a name, is transmitted to the web server, even as the IP layer uses the numeric address returned by DNS. This decouples the service from an IP address.

The 32-bit IPv4 address was more than adequate for the ARPANET. It would have been sufficient forever. IPv6 "fixes" the shortage of addresses by putting a 128-bit source and destination address in every packet. This simply perpetuates IP's architectural flaws. One of the more surprising results from the theory underlying RINA is that a global address space is unnecessary. Limits of space do not permit a full explanation, but it fairly apparent that a 32-bit address is more than enough for any layer.

In RINA, applications are accessed by name, and only by name. An application name names the application, not a path to it as in the current Internet. Since a DIF is a Distributed Application that does IPC, they also have names.

The members of a DIF are application processes (referred to as IPC Processes) and hence have application names. Addresses are simply synonyms for IPC Processes taken from a name space whose scope is limited to the DIF, and may be structured to make them more useful within the DIF. Addresses are only known by members of the DIF, not the user applications. Since addresses are often location-dependent with in the DIF, they must be assigned by the DIF - it is the only one who knows where they are. Since addresses name where the relaying is done, multi-homing is a consequence of the structure. Mobility is obviously nothing more than dynamic multi-homing, i.e. changing points of attachment more often than usual (see Figure 4).

UNICAST, MULTICAST, AND ANYCAST

RINA names are more flexible than IP addresses. The most obvious form of reference, of course, is unicast, an application resident in a single place. No known network architectures name hosts. There is a sloppy use of language that gives this impression but it is not true.

An application can live in one place, which is addressed via unicast. But often an application resides on more than one host. Popular web sites are redirected, invisible to the user. In so doing, they are simulating a second type of address, anycast. Any one of a set of destinations can serve the request. (Once a dialog is established between two hosts this way, it is likely to be completed via unicast.) The third case is multicast, in which the information is relayed to every member of a set of destinations.

Multicast and anycast turn out to be subsets of a single form, whatevercast. Both deal with a set of addresses and a rule. Anycast addresses one member while multicast returns all members that satisfy the rule. Hence unicast can be seen as the special case, one where the set has a single member. Thus applications need do nothing special to deal with multicast or anycast. Just how a DIF implements whatevercast is an internal matter, of course, since it's a black box.

The value of efficient multicast is obvious in the case of streaming video, which can be sent to a set of destinations. This set can be dynamic: When a user wants to watch the stream, the set top box, smartphone or computer is immediately enrolled in the multicast set. Only one copy of the stream is relayed to any given DIF, though that DIF may use its multicast capability to relay the stream to as many other DIFs as necessary to reach the members of the set. This is far more efficient than IPTV in which

" Adopting RINA should be easier than following the IETF's recommendations to transition from IPv4 to IPv6. "

every viewer is watching its own stream from a server. It could even provide a practical way for cable TV to evolve to a common structure with the Internet itself.

EASIER ADOPTION THAN IPV6

RINA can only be of practical use if it can work with TCP/IP, at least initially. In fact, adopting RINA should be easier than following the IETF's recommendations to transition from IPv4 to IPv6. That's difficult at best. IPv6 was not designed for backward or forward compatibility with its predecessor. They occupy the same place in a fixed protocol stack.

RINA provides more options for its phased adoption. It is not fixed to one place in the protocol stack, so implementations can be flexible. It treats TCP/IP as a sort of limited DIF. RINA thus can be applied below TCP/IP, to provide network service to TCP/IP hosts, or to connect IP networks. RINA can be used above IP, using existing IP links as a transport medium. And RINA networks can be gatewayed to TCP/IP networks, translating at least some applications between the two.

Bottom line: The adoption of RINA is seamless. Internet applications on a RINA DIF can be accessed transparently from the Internet and vice versa.

RINA, as it becomes available, can thus gradually supplant TCP/IP. A backbone network could be built using RINA, supporting both types of upper layers. And RINA-native applications could be developed and rolled out. Since RINA does not depend on globally-unique IP addresses, a single IP address could function as a gateway to a RINA network in which applications communicate using names.

SUMMARY OF BENEFITS

It can be seen that RINA offers a number of benefits compared to TCP/IP. These include improved scalability (the ability to efficiently support larger networks) and security (both privacy of communications and protection against attack). It addresses the "3M" challenges of mobility, multicasting and multi-homing. It provides a standard mechanism for application development. It solves the neutrality issue by providing QoS options without allowing deep packet inspection or even making the application visible to an underlying network. It provides an easy adoption path for IPv4 users. And it does all this with a radical simplicity that will facilitate lower cost, higher-efficiency implementations. By returning to the fundamentals and recognising the patterns in network architecture, RINA promises to move networking to the next level.

AFTERWORD FROM JOHN NOLAN

Goldstein and Day's paper is but a brief introduction to RINA, with Day's book an excellent read should the reader wish to delve further[2]. One of the questions that the work of the Pouzin Society raises for me is whether there is another " contender"[3] but I'm not so sure if there is? There is no doubt that the TCP/IP protocol suite is now fully established and as an example "IPv6 day[4]" has been and gone [3]. Continuing on the lack of alternatives and in a recent edition of the aforementioned IP Journal [4], Geoff Huston (co-editor for the edition) states under "Myth 8" that "no viable substitutes exist." I'm guessing that the principal authors of this paper may disagree with that statement?

So is the TCP/IP protocol suite dancing its Last Waltz and if so, is RINA the only alternative? I'm keeping an open mind (as I don't know the answer), but I would encourage readers to get involved, particularly as the concepts underpinning RINA has certainly challenged my thinking. Perhaps we are too heavily entrenched with the TCP/IP protocol suite to have an open and honest debate - time will tell?

ABBREVIATIONS

ARPANET	Advanced Research Projects Agency Network
CDAP	Common Distributed Application Protocol
DIF	Distributed IPC Facility
DNS	Domain Name System
DTCP	Data Transfer Control Protocol
DTP	Data Transfer Protocol
EFCP	Error and Flow Control Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPC	Interprocess Communications
ISO	International Standards Organisation
MPLS	Multi Protocol Label Switching
NAT	Network Address Translation
NCP	Network Control Program
OSI	Open Systems Interconnection
POA	Point Of Attachment
QoS	Quality of Service
RINA	Recursive Internetwork Architecture
TCP	Transmission Control Protocol

FOOTNOTES

1 This paper has been abridged by Fred Goldstein from the original paper, which can be found at [5]

2 If only to relive some of the historical moments and to gain an insight into the political discussions during the "protocol wars."

3 During the protocol wars, I can recall OSI vs. TCP/IP vs. SNA with Decnet another player at the time.

4 For a European perspective on IPv6 day see the comments from RIPE [6].

References

1. http://www.cisco.com/web/about/ac123/ac147/about_cisco_the_internet_protocol_journal.html. [Online]
2. http://www.pouzinsociety.org/. [Online]
3. http://www.worldipv6day.org/. [Online]
4. http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_14-1/141_myths.html. [Online]
5. http://pouzin.pnanetworks.com/images/PSOC-MovingBeyondTCP.pdf. [Online]
6. http://labs.ripe.net/Members/emileaben/measuring-world-ipv6-day-glitches-and-lessons-learned. [Online]

ABOUT THE AUTHORS

JOHN DAY

John Day has been involved in research and development of computer networks since 1970, when they were the 12th node on ARPANET and has developed and designed protocols for everything from the data link layer to the application layer. Also making fundamental contributions to research on distributed databases. He managed the development of the OSI reference model, naming and addressing, and a major contributor to the upper-layer architecture. He was a major contributor to the development of network management architecture, working in the area since 1984 and building and deploying a network management system, a decade ahead of comparable systems. John has published Patterns in Network Architecture: A Return to Fundamentals, which has been characterised (embarrassingly) as “the most important book on network protocols in general and the Internet in particular ever written.” John is also a recognised scholar in the history of cartography, and contributed to exhibits at the Smithsonian. He can be contacted at jeanjour@comcast.net

FRED GOLDSTEIN

Fred Goldstein advises companies on technical, regulatory and business issues related to the telecommunications, cable, wireless and Internet industries, especially in areas where they overlap. He assists service providers in network design, business modelling, planning, and technical architecture. He has frequently been an expert witness in intercarrier compensation and network interconnection cases. He has worked with enterprise networks on a wide range of matters such as backbone network design, voice systems planning, and traffic engineering. The author of numerous articles including The Great Telecom Meltdown and ISDN In Perspective, he has served on standards committees in areas such as ATM networks and Frame Relay, and has taught courses for Northeastern University and National Technological University. He can be contacted at fgoldstein@ionary.com

JOHN NOLAN

John Nolan has been active in the telecoms industry since 1971 and after leaving BT in 1979 he spent extensive periods in senior telecom roles in both Local Government (GLC/ILEA) and subsequently, the Financial Services industry (NatWest Bank Group). In 1995, he joined Intercai Mondiale (www.intercai.co.uk) where he was lucky enough to enjoy many differing cultures whilst consulting in Europe, the USA, and the Middle and Far East. In 2003, he left Intercai Mondiale and co-founded First Mile Networks (www.firstmilenetworks.co.uk) with Dr Chris Lilly. John has a degree in Electrical and Electronic Engineering from London South Bank University and is a member of the Institution of Engineering and Technology; a member of the Institute of Electrical and Electronics Engineers; and is also a member of the Association for Computing Machinery. He can be contacted at john.nolan@firstmilenetworks.co.uk.

Call for Papers

Authors are encouraged to submit articles for publication in The Journal.
Interested parties should e-mail journal@theITP.org in the first instance with topic suggestions.