Detecting Duplex Mismatch on Ethernet

Stanislav Shalunov and Richard Carlson*

{shalunov,rcarlson}@internet2.edu

Abstract. IEEE 802.3 Ethernet networks, a standard LAN environment, provide a way to auto-negotiate the settings of capacity (10, 100, or 1000 Mb/s) and duplex (full- or half-). Under certain conditions described below, the auto-negotiation protocol fails to work properly. The resultant configuration problem, *duplex mismatch*, appears to be common; when this problem occurs, the connectivity is impaired, but not completely removed. This can result in performance problems that are hard to locate.

This paper describes a work in progress aimed at (i) studying the condition of duplex mismatch in IEEE 802.3 Ethernet networks, (ii) producing an analytical model of duplex mismatch, (iii) validating the model, (iv) studying the effects of duplex mismatch on TCP throughput, (v) designing an algorithm for duplex mismatch detection using data from active testing, and (vi) incorporating the detection algorithm into an existing open-source network troubleshooting tool (NDT).

1 Introduction

Ethernet duplex mismatch is a condition that occurs when two devices communicating over an IEEE 802.3 Ethernet link (typically, a host and a switch) do not agree on the duplex mode of the direct Ethernet connection between them; the switch might operate as if the connection were full-duplex with the host operating in half-duplex mode, or *vice versa* (section 2 describes the condition in more detail). Duplex mismatch causes some packets to be lost (see section 3) and, in many cases, leads to serious performance degradation (see section 4.2). Section 5 discusses soft failures similar to duplex mismatch. Section 6 describes an algorithm to detect a duplex mismatch condition by observing the behavior of the network when test traffic is injected, and describes integrating the detection algorithm into our existing network testing system.

2 Problem description

The IEEE 802.3 Ethernet specifications [1] define a set of Media Access Control (MAC) protocols that are the standard for Local Area Networks (LANs) used around the world. The original protocol was developed for communications over

* This work was supported by contract 467-MZ-401761 from the National Library of Medicine.

shared media and uses the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol to arbitrate access to the shared media. That the media is shared means that a host could either transmit or receive data at any given time (in other words, devices operate in a *half-duplex mode*). Later enhancements define how devices can communicate over a wide variety of co-axial, twisted-pair, and fiber-optic cables. The twisted-pair and fiber-optic cable protocols allow for development of switched network environments—enabling a device to simultaneously transmit and receive data (thus, operating in a *full-duplex mode*). All IEEE 802.3-compliant implementations (10-, 100-, and 1000-Mb/s) must support half-duplex operations and may support full-duplex operations.

2.1 Half-duplex Ethernet operation

To reliably operate in the half-duplex mode ([1], section 4.1), it is necessary to specify how a device should operate when several devices attempt to transmit simultaneously (a *collision event* is said to occur when this happens). The CSMA/CD protocol is used to resolve collisions: in half-duplex mode, an interface will monitor the Carrier Sense (CS) flag at all times to determine if any device is transmitting data on the network; when the local device has data to send, these data pass through one or more upper-layer protocols and finally arrive at the local interface layer where they are encapsulated into an Ethernet frame. The transmitting interface checks the CS flag and, if it is *true*, holds the frame in the transmit queue. When the CS flag becomes *false*, an inter-frame gap (IFG) timer is started. When this timer expires, the frame is transmitted onto the network. While the frame is being transmitted, the interface monitors the outgoing data to detect if another device has also started to transmit. This Collision Detection (CD) algorithm improves network efficiency by stopping transmissions when they are obviously corrupted. The CSMA/CD protocol also prescribes a way to retransmit frames after the collision is detected.

Two types of collisions may be detected by the transmitting interface. A slot time is defined as the time it takes to transmit the first 64 octets (512 octets for 1000-Mb/s interfaces) of the frame. A normal collision will occur within the slot time period, a *late collision* will occur from the end of the slot time to the end of the frame. If no collision is detected within the slot time, the transmitting interface is said to have captured the network. If a normal collision is detected, an interface will halt its transmission, broadcast a jamming signal to ensure the collision is detected by other devices, and then wait a pseudo-random time before attempting to transmit again. A host that detects a late collision will still stop transmitting its frame and discard the incoming frame. At speeds up to 100 Mb/s, the standard does not make it mandatory to retransmit the outgoing frame (this is implementation-specific); at 1000 Mb/s, the frame shall be discarded. Late collisions should not occur on a properly operating network.

2.2 Full-duplex Ethernet operation

With the introduction of the 10BASE-T specification for 10-Mb/s twisted-pair media ([1], section 14.1.2), full-duplex operation over switched network connections became possible. In switched mode, a network link capable of transmitting and receiving data simultaneously is dedicated to each device in the network. This usually means that two twisted-pair copper or fiber-optic cables are used create a bidirectional network link. This operating mode offers several advantages over half-duplex operation: a device can transmit data whenever they are available, thus improving performance; in addition, the need for CSMA/CD vanishes, thus simplifying the operation and making it ostensibly more robust.

When a device has data to send in full-duplex mode ([1], section 4.1), an Ethernet frame is created and placed in the transmit queue. The frame at the head of the queue is transmitted after the expiration of the IFG timer. Since collisions are impossible, a higher throughput can be achieved; the interface can also stop monitoring the CS and CD flags.

2.3 Auto-configuration on Ethernet

In addition to operating in full- or half-duplex mode, the IEEE 802.3 specifications ([1], section 28.2) describe how a device can configure itself to operate in an unknown environment by use of a negotiation protocol that detects the capacity and duplex settings. The protocol uses an out-of-band pulse-code sequence based on the 10BASE-T link integrity test pulses. This string of closely spaced pulses, the Fast Link Pulse (FLP) burst, encodes the operating modes that each interface is capable of supporting into a Link Code Word (LCW).

The protocol operates thusly: at power-on, after a physical cable is connected, or after a management command is received, the interface enters the auto-negotiation phase. At this time, the interface starts sending FLP bursts to the remote interface and listening for incoming FLP bursts. An old 10BASE-T-only interface will begin sending only single Normal Link Pulses (NLP), indicating it does not support the auto-negotiation function. The receiving interface measures the interval between NLP pulses to differentiate between the NLP and FLP bursts. If the FLP bursts are detected, then the Auto-Negotiate flag in the Media Independent Interface (MII) control register is set to *true* and the auto-negotiation process continues. If the FLP bursts are not detected, auto-negotiation is disabled and the interface is brought up in its default state.

The default parameter settings ([1], section 22.2.4) determine how the interface will operate if auto-negotiation fails (the interface will set the interface to the highest possible speed and half-duplex).

Upon receipt of three consecutive and consistent LCWs, the receiving interface will respond by setting the ACK bit in the transmitted LCWs and the MII control register to *true*. The receipt of three consecutive and consistent LCWs with the ACK bit set indicates that the peer interface has received the negotiation parameters and is ready to complete the negotiation process. The mode priority ([1], annex 28b.3) settings determine how the auto-negotiation process

determines which speed and duplex setting to use. This process prefers the highest performance capability that both interfaces support, so 100 Mb/s is preferred over 10 Mb/s and full-duplex is preferred over half.

In addition to the auto-negotiation protocol, an auto-sense protocol ([1], section 28.2.3.1) can detect 100BASE-TX, 100BASE-T4, and 10BASE-T interfaces that do not support the auto-negotiation protocol. In this situation, an interface that receives an NLP signal will use the link integrity test functions to detect the speed of the peer interface. While speed and media type (TX or T4) is detected, only half-duplex operations are supported by this protocol.

Some older (or cheaper) network interface cards support only half-duplex mode; most modern cards support both auto-negotiation and manual configuration modes. It should be noted that the speed and duplex settings chosen by an interface are never explicitly communicated to the peer interface. We believe this lack of robustness makes the auto-negotiation protocol susceptible to errors that can cause operational problems on real networks.

2.4 Duplex mismatch

While auto-configuration makes connecting to the network easier, it can lead to major performance problems when it fails, as it sometimes does [2]. If the two sides of a Fast Ethernet connection disagree on the duplex mode (*i.e.*, one is using full- and the other is using half-), a *duplex mismatch* is said to occur.

A duplex mismatch can happen in one of these ways (among others), but the symptoms seen by the hosts will be the same regardless of the cause:

1. One card is hard-coded to use full-duplex and the other is set to auto-negotiate: the hard-coded side will not participate in negotiation and the auto-negotiating side will use its half-duplex default setting;
2. The two cards are hard-coded to use different duplex modes;
3. Both cards are set to auto-negotiate, but one or both of them handles auto-negotiation poorly [3, 4]; note that, in this case, the problem can occur sporadically and rebooting or resetting the interface on either side could clear the condition.

3 Predicted behavior

When duplex mismatch happens, a peculiar breakdown in communication occurs. Denote the interface that thinks that the connection is in full-duplex mode F and the interface that thinks that the connection is in half-duplex mode H.

3.1 Model of the pattern of layer-3 packet loss

Denote the propagation delay between H and F by δ (in seconds),[1] the period of time after starting to send during which H will retransmit in case of collision

[1] The standard requires that cables not exceed 100 m in length; this means that $\delta \leq 0.5\,\mu$s for 100-Mb/s Ethernet.

by ξ (in seconds),[2] and the capacity of the link by c (in bits/second). Let us consider the two directions separately.

F → H Duplex mismatch can cause loss in this direction (but no extra delays can be introduced, as F never delays frames in its transmit queue and never retransmits frames). Suppose that a frame is being sent in this direction at time t_F. It will be lost if and only if it starts arriving at H while H is sending: *i.e.*, there exists a frame of size m (in bits) sent by H at time t_H such that

$$t_H < t_F + \delta < t_H + m/c. \tag{1}$$

H → F Duplex mismatch can cause loss in this direction for three reasons:

(i) Non-recoverable collision loss. Suppose a frame of size m (in bits) is being sent by H at time t_H. This frame will be lost if and only if there exists a frame sent by F at time t_F such that

$$t_H + \xi < t_F + \delta < t_H + m/c. \tag{2}$$

(ii) Buffer overrun loss. If the link $F \to H$ is not idle long enough for H to drain its transmit queue, the queue will overflow and packets will be lost. If the average rates at which F and H are sending are c_F and c_H, loss will always happen if

$$c_F + c_H > c. \tag{3}$$

Note: Since H can spend considerable time idle due to exponential back-off,[3] buffer overrun loss can occur even if condition 3 does not hold.

(iii) Excessive collision loss. A frame will be lost if a collision occurs more than 16 times when H attempts to retransmit the frame.

In addition, delays can be introduced by H waiting for the network to go idle before sending and by retransmitting. The more traffic goes in the $F \to H$ direction, the more traffic going in the $H \to F$ direction will be delayed.

3.2 Manifestation of duplex mismatch in the case of UDP

Using UDP *per se* imposes no particular sending schedule. For the purposes of producing verifiable predictions made by our model, the case of Poisson streams is considered in this section; this case is easy to analyze and, therefore, the verification of the predictions will help validate the model.

[2] Note that ξ is determined by a specific interface card model. The standard guarantees that for 10- and 100-Mb/s Ethernet, $\xi \geq 512/c$; for 1000-Mb/s Ethernet, $\xi = 4096/c = 4096/10^9 \approx 4.1\,\mu$s.

[3] For attempt n, the delay is a uniformly distributed pseudo-random integer between 0 and $2^{\min(n,10)} - 1$, multiplied by slot time.

Assume that two Poisson streams of frames are injected into the system for transmission by the interfaces. The average rate of the streams are c_F and c_H (in bits/second, as above). Since F never delays packets, the stream that leaves F is Poisson. Let us consider the situation when $c_F \ll c$ and $c_H \ll c$. In this case, the stream leaving H is not disturbed to a large extent and can be approximated by a Poisson stream.

Our model then predicts loss in $F \to H$ direction, p_F, to be

$$p_F = \frac{c_H}{c}. \tag{4}$$

In the $H \to F$ direction, since $c_F + c_H \ll c$, buffer overrun loss will never occur (cf. condition 3). Excessive collision loss rate, $(c_F/c)^{16}$, will be negligible. Further, denote the size of packets that leave H by m. We have:

$$p_H = \frac{c_H}{c} \max\left(0, 1 - \frac{c\xi}{m}\right). \tag{5}$$

Note: Formula 5 allows one to measure ξ externally by observing $\max(0, 1-c\xi/m)$ (the proportion of bits in frames sent by H that are transmitted later than ξ seconds after start of frame transmission).

3.3 Manifestation of duplex mismatch in the case of TCP

TCP is a reliable transport protocol that uses ACK packets to control the flow of data between two Internet nodes. Only unidirectional TCP data flows with ACKs flowing in the opposite direction are considered. Denote the interface on whose side the TCP sender is located S, and the interface on whose side the TCP receiver is located R.[4] In bulk transfer mode, a stream of MTU-sized packets will go from S to R and a stream (with typically half as many packets per second when delayed ACKs are used) of small packets containing TCP ACKs will go from R to S. ACKs are cumulative: if an ACK is lost and a subsequent ACK is delivered, the same information is conveyed (and the effect of the missed increment of the TCP congestion window is minimized). Denote the period of time it takes the receiver to generate an ACK, and for this ACK to reach interface R, by Δ.

Since, with TCP, packets flow in both directions, there is a potential for collisions and loss during normal TCP operation on a link with duplex mismatch. Consider the case when TCP is not in a timeout state and congestion window is not too small (at least a few packets). If the network path between the sender and the receiver does not contain any tight links, then the arrival of several back-to-back TCP data packets should cause a collision and a loss event will occur.

[4] Note that S and R are *not* the sender and receiver, but rather the two interfaces on the sides of a link with duplex mismatch. Often, S could be on the sender or R could be on the receiver, but for both to be true, the network path would need to consist of exactly one layer-2 hop.

For simplicity, consider two cases where a single duplex mismatch condition exists on the last hop of the network path (*e.g.*, next to the user's computer):[5]

1. $S = F$, $R = H$: The interface R will obey the CSMA/CD protocol and refrain from transmitting while a frame is being received. It will also detect and re-transmit frames when collisions occur using the proper collision slot time (defined in 2.1). The interface S will follow the full-duplex protocol and transmit frames whenever they become available without checking for frames on the receive circuit. Collisions will be ignored by S and the entire packet transmission time m/c will always be used.
   When gaps between data packets are wider than Δ, ACK packets will be transmitted and will arrive at the sender for processing by the TCP stack. When gaps between data packets are less than Δ, a collision could occur. Consider the case during slow start when the congestion window (CWND) on the sender reaches four packets; the sender could have four data packets to transmit back-to-back. After receiving two data packets, the receiver would generate an ACK. The interface R will receive this ACK in time Δ, attempt to transmit it to S, and find that S is currently transmitting the third data packet, so R will delay the ACK's transmission. When this frame's transmission completes, both R and S will wait an IFG time and simultaneously begin to transmit (the fourth data packet and the ACK packet) causing a collision to occur. R will detect this collision and re-schedule another retransmission, but S will continue to blindly send the entire frame. This frame will be discarded by R as corrupted due to the collision. The sender will detect and handle this loss event.
   In general, whenever CWND on the sender increases enough to allow for a burst of at least four packets to arrive at the receiver, the ACK generated in response to the first or second data packet in the burst will allow the next packet to be delivered, but will cause all subsequent packets in the burst to be lost; the ACK itself will be delivered to S after the burst. TCP would thus suffer from both inability to raise CWND and timeouts caused by multiple packet losses in a single RTT (whenever CWND becomes large enough). Empirically, TCP infrequently enters slow start in this case, since CWND remains small enough; the goodput obtained is thus better than that in case 2.
   The large number of lost data packets will cause the receiver to generate a large number of duplicate ACKs. Thus, the TCP source node will receive a larger number of ACK packets than would normally be expected in a delayed ACK environment.
2. $S = H$, $R = F$: Consider the case where a burst of packets arrives at S so that the next frame is ready to begin transmission before the previous frame ends. The first two data packets will arrive at the receiver, which will generate an ACK packet. This packet will be transmitted during the receipt of the third data packet if Δ is small enough. The switch will detect the

[5] Only data and ACK packets from a single TCP flow are considered.

collision and abort the data transer, but the ACK packet will be lost. If $\Delta < \xi$, then the data packet will be resent by S; otherwise, it will be lost. This loss of ACKs not only has a detrimental effect on TCP throughput by reducing the number of increments of CWND, but also creates a situation when the last ACK for a burst is not received, thus causing the sender to timeout and reenter slow start.
The large number of lost ACK packets will mean that the sender will see fewer ACKs then would otherwise be expected (about one ACK per burst).

4 Validation of predictions

To validate the model of section 3, we create the duplex mismatch condition artificially, send test traffic, and compare the results with predictions.

4.1 Validation with UDP streams

Since the loss pattern created by duplex mismatch is complex, it is easier to analyze first the results with simple synthetic traffic consisting of UDP packets. Our model makes the following predictions (see section 3.2):

1. Unidirectional traffic (in either direction) will not suffer any loss;
2. When a small amount of traffic is sent in each direction as a Poisson stream, loss is given by formulae 4 and 5.

Our UDP tests were run with a program THRULAY (using option `-u`), which had UDP mode added for the purposes of conducting these experiments.[6] A duplex mismatch condition was artificially created between a Linux host and an Ethernet switch with MII-TOOL. A series of bidirectional 10-second UDP tests were conducted between the host with the duplex mismatch condition and another host connected to the same Ethernet switch. Both hosts were connected at 100 Mb/s. The host without duplex mismatch was connected using full duplex. The sending rate was varied from 1 Mb/s to 9 Mb/s on each side in 1-Mb/s increments; 81 tests were run.

The results of these experiments are presented in figures 1 and 2. In figure 1, the line predicted by equation 4, on which data points are expected to fall, is shown; as can be seen, the match is quite good. In figure 2, data points for any given value of c_H are expected (in accordance with equation 5) to fall on a horizontal line, which generally agrees with the observations. Prediction 2 thus holds. In might be of interest that indirectly, the value of ξ (inherent in the Ethernet implementation of the HP switch we used in this experiment) is

[6] Many network testing programs allow the use of a UDP mode. However, we found no program that would generate a Poisson streams of packets that are sent with no correlation with operating system's time slice boundaries. Our tool can use a busy-wait loop on the TSC timing register to effect a true Poisson stream. The program is made freely available at `http://www.internet2.edu/~shalunov/thrulay/`.

measured here and it appear that the value is quite small—perhaps as small as allowed by the standard.[7]

In addition, we conducted tests in each direction without opposing traffic; no loss was observed during these tests, thus verifying prediction 1.

Formula 4 was only proven for $c_H \ll c$ and $c_F \ll c$. However, empirically (data not presented in this paper), this formula extends to values of c_F/c as large as 0.3 or even 0.4 and values of c_H/c as large as 0.5.

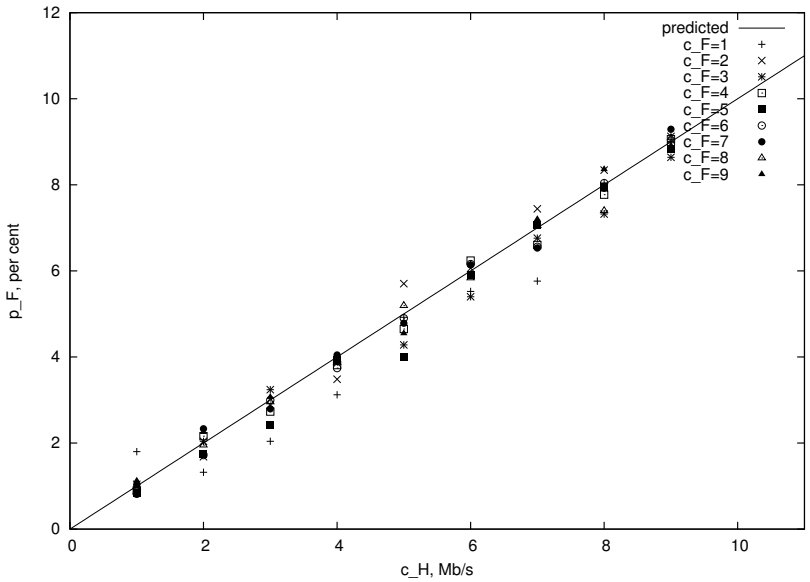

Fig. 1. Loss probability p_F in the stream in $F \to H$ direction as a function of c_H, the rate of the stream going in the opposite direction. The prediction is based on equation 4.

4.2 Validation with TCP streams

To validate the model developed in section 3.3, two types of tests were run. One test used a pseudo-TCP stream consisting of a stream of UDP packets sent back to back, with a small stream returning in the opposite direction; the other involved actual TCP streams.

The TTCP test program was modified to perform these experiments. The modifications involved having one host send a stream of 50 UDP packets to

[7] The best match for the value of the term $\max\left(0, 1 - \frac{c\xi}{m}\right)$ is about 0.93. For these tests, we had $m = 1518\,\mathrm{B}$; therefore, ξ was large enough to cover about 106 B of each packet.

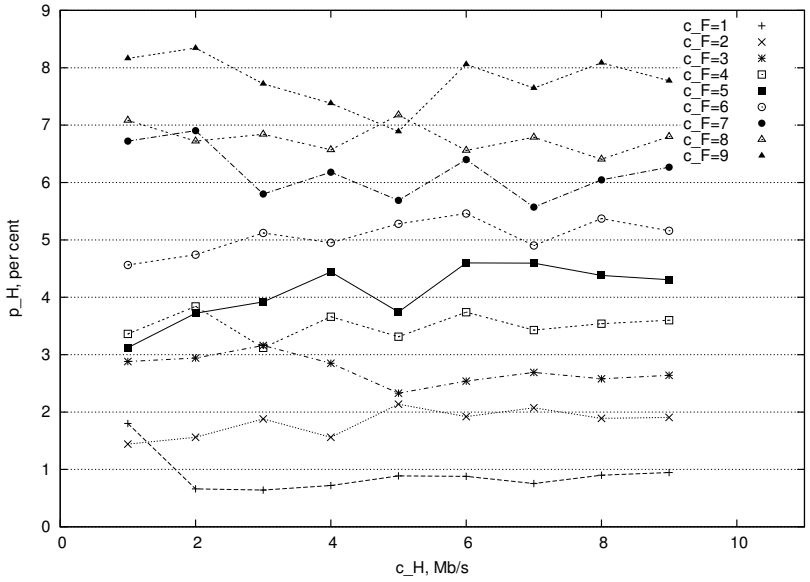

Fig. 2. Loss probability p_H in the stream in $H \rightarrow F$ direction as a function of c_H, the rate of the stream.

another. Each packet contained a unique sequence number. The receiver was modified to return a small UDP packet for every second packet received. This simulates the typical mode of operation of the delayed ACK algorithm found in TCP implementations. This returned packet contained a unique sequence number and a copy of the data packet sequence number that generated this ACK. As before, the MII-TOOL program was used to force one link into different normal and mismatch states. The raw data was captured using the TCPDUMP command.

An analysis of the resulting traces showed that there were two different mismatch behaviors and two flavors of each behavior. These behaviors match the predicted models described in section 3.3.

One behavior is when $S = H$ and $R = F$. In this situation, most of the returning ACK packets are lost. Even though the receiver is generating and transmitting them, only the final ACK packet is recieved by the sender. The rest are lost due to collisions with the arriving data packets. Depending on how late collisions are handled, the data packets may be retransmitted or discarded. Thus, two flavors of this behavior are observed.

The second behavior is when $S = F$ and $R = H$. In this situation, the arriving data packets are discarded due to collisions. The ACK packets are delayed due to numerous retransmission attempts. When the receiver's local link is mismatched, Δ is small, so the first ACK is given back to the interface while data packet 3 is being received. As predicted in section 3.3, data packets 4, 5, 6, 7 and 8 are lost

due to collisions as this frame is retransmitted over and over. Eventually, the retransmit delay increases to a point where several data packets do get through, causing more ACKs to be generated. These ACKs are queued behind the first ACK and can only be transmitted after a successful transmission (or if the maximum retransmission count is exceeded).

A slightly different flavor of this behavior occurs when the mismatch link is not the last link in the path. In this case the first collision will occur when the ACK has propagated back along the path to the point of the mismatch. In the tests, the sender and receiver roles were exchanged but no changes were made to the link with duplex mismatch. This resulted in data packets 6, 7, 8, 9, and 10 being lost. This shift exactly matches the extra time needed for the ACK to propagate back along the network path to the mismatched switch port.

For comparison purposes, the normal full-duplex and half-duplex operating conditions were also tested. No exceptional loss or delay values were recorded.

While this simple test can demonstrate how the returning ACK packets can collide with data packets, it does not explain the complex dynamics of a real TCP flow. We need to capture the retransmission and congestion control dynamics to more completely describe the TCP behavior.

To examine the various normal and mismatch cases, a series of tests were run using the existing NDT client/server program. The client's link was configured using the MII-TOOL program to simulate the expected configuration: where the NDT server is properly configured but the client may have a mismatch condition. The results of these test are described below.

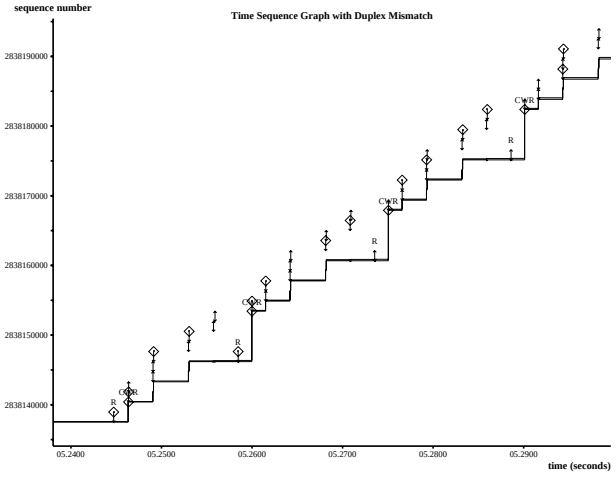

Fig. 3. TCP trace with duplex mismatch, $S = F$ and $R = H$

Figure 3 shows the mismatch case with $S = F$. We begin the trace after the NDT server has successfully sent several packets and a timeout occurred. Following the retransmission, an ACK is received causing the RWIN value to increase and allowing more data to be sent. Two data packets are transmitted, and, following an RTT, the ACKs are returned. Note that after two RTTs a hole appears due to a lost packet. The sender continues to inject new packets as ACK arrive. After three duplicate ACKs have arrived, the sender retransmitts the missing data packet. The receipt of this packet fills in the hole and causes a large jump in RWIN. This process repeats for the entire 10 second test time.

The result is that throughput is limited by CWND and the RTT of the end-to-end path. Slight variations in packet arrival times can cause multiple packets to be lost at the client. This increases the probability of a TCP RTO event, further decreasing in throughput.

When the mismatch condition is reversed such that $S = H$, an even greater impact is seen on TCP throughput. As noted above, this condition will result in ACK packets being discarded. A typical trace shows that the NDT server sends several back-to-back packets to the receiver. These packets are received, but the returning ACK is lost. The lack of an ACK prevents the sender from sending more packets into the network, thus preventing any more ACKs from being generated. The connection stalls until the TCP RTO event occurs. This causes the sender to resend what it thought were lost packets. When the first retransmitted data packet is received, the receiver recognizes it as a duplicate and immediately sends a duplicate ACK. This ACK causes two more packets to be sent, another duplicate and a single new packet. When the receiver receives the duplicate data packet, it sends another duplicate ACK, which collides with the third data packet. Thus, the ACK is lost, but the data packet is successfully received after a small delay. The loss of this ACK causes another RTO event to occur, starting the entire process over again. TCP throughput in this situation is limited by the numerous RTO events.

5 Soft failures similar to duplex mismatch

Our goal is to develop a duplex mismatch detection algorithm that treats the network as a black box. Both false positives and false negatives must be considered. A particularly harmful and insidious mode of false positive would be the characterization of another soft failure condition as duplex mismatch: not only would users waste their time not solving their real problem, but they might, while responding to perceived duplex mismatch diagnostics, change settings to introduce duplex mismatch in a network that previously did not have one. For example, when a copper twisted-pair cable is subtly damaged, *cross-talk* can occur; *i.e.*, a signal transmitted on one wire induces a spurious signal on another. Bit corruption during bidirectional transmission that occurs thusly could be confused with duplex mismatch. Chief differentiators of cross-talk from duplex mismatch are:

1. Cross-talk can affect unidirectional traffic; duplex mismatch cannot;

2. Duplex mismatch occurs deterministically; cross-talk corrupts bits randomly.

6 Detecting duplex mismatch in NDT

While other packet losses can cause the number of duplicate ACK packets to increase, the asymmetric throughput results are not observed. Thus we believe that the combination of these two conditions is a reliable indicator of duplex mismatch.

The Network Diagnostic Tool (NDT) [5] is an open-source software program that uses a client/server approach to detect common network problems. The server process communicates with a Java-based client program to generate a series of TCP unidirectional data flows. The server also captures Web100 data allowing it to delve into the depths of the TCP connection. By analyzing this data, it is possible to determine if a duplex mismatch condition existed over some portion of the network path.

The original NDT duplex mismatch detection heuristic was created after running a series of experiments in a testbed network environment. This environment was expanded to encompass a campus network. This heuristic used the amount of time the connection spent in the congestion window limited case, the number of packets retransmitted per second, and the throughput predicted by the $\mathrm{MTU}/\mathrm{RTT}\sqrt{p}$ formula. A later modification was made to ignore cases when the client was located behind a tight link (cable modem or DSL circuit).

A new algorithm that takes advantage of the analytical model described in this paper has now been incorporated into the NDT tool. Web100 [6] variables can be used to perform duplex mismatch detection thusly: The NDT server generates two test streams sequentially, one on each direction, to measure TCP throughput. Each stream is monitored to determine the path capacity. Thus, we have two TCP throughput measurements and an estimate of the tight link in the end-to-end path.

Duplex mismatch causes major disruptions to a TCP flow. The generation and transmission of ACK packets in direct response to received data packets increases the probability of a collision. Once this happens, either the data or the ACK packet will be lost and the other delayed by the CSMA/CD algorithm. However, TCP is a reliable transport protocol, so it will retransmit lost data packets. These retransmission will cause the receiver to generate more duplicate ACKs per data packet compared to one delayed ACK for every other data packet.

The Web100 variables are captured on the NDT server only in the case where the NDT server is sending data to the client. The two duplex mismatch cases can now be examined.[8]

If the client is the receiver and the mismatch condition is such that $R = H$, then numerous data packets will be lost. The original transmission and an subsequent retransmissions will cause the Web100 DataPktsOut counter to increment. In addition, the loss of individual packets will cause the client to generate a large

[8] In each case, the TCP flow in the opposite direction will exhibit the other mismatch behavior.

number of duplicate ACKs. The Web100 AckPktsIn counter will be incremented every time a new or duplicate ACK is received. Thus, the ratio of data packets transmitted *vs* ACK packets received will skew away from 2:1 towards more ACKs. As noted above, throughput will be a function of CWND and RTT.

If the client is the receiver, $R = F$, and $S = H$, then numerous ACK packets will be lost. In addition, a large number of packets will be retransmitted and a large number of timeouts will occur. This will skew the data packet *vs* ACK packet ratio in the opposite direction from that described above. Thus, a ratio of more than 2:1 is expected. TCP throughput will be dramatically affected due to the large number of RTO events.[9]

This means that we can create a reliable duplex mismatch detection algorithm by combining the asymmetric throughput with the skewed ACK:data packet ratio.

At present, we are collecting data to validate these predictions and results. NDT servers at several locations are gathering data from production environments. We will analyze the log files produced and compare the results with the observations from the NDT administrator. Our presentation will describe the results of this effort.

7 Conclusions

Duplex mismatch is an insidious condition, undetectable with such simple network operation tools as *ping*. It can affect a user's performance for weeks or months before it is detected. A model of duplex mismatch is described and a detection algorithm is proposed. The algorithm is implemented in the NDT.

References

1. IEEE: Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications. IEEE Standards, 802.3 (2002)
2. Apparent Networks: Duplex conflicts: How a duplex mismatch can cripple a network. White Paper (2002)
3. Eggers, J., Hodnett, S.: Ethernet autonegotiation best practices. Sun BluePrints OnLine (2004)
4. Hernandez, R.: Gigabit Ethernet auto-negotiation. Dell Power Solutions **1** (2001)
5. Carlson, R.A.: Developing the Web100 based network diagnostic tool (NDT). In: Proc. Passive and Active Measurement Workshop (PAM), San Diego (2003)
6. Mathis, M., Heffner, J., Reedy, R.: Web100: Extended TCP instrumentation for research, education, and diagnosis. ACM Computer Communications Review **33** (2003)

[9] The NDT tool will display both conditions as it runs two unidirectional tests.