Windowed Ping: An IP Layer Performance Diagnostic DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT

Matthew Mathis <mathis@psc.edu>

Abstract

The Internet is suffering from multiple effects of its own rapid growth. Network providers are in the uncomfortable position of deploying new products and technologies into already congested environments without adequate tools to easily assess their performance. In this paper we present a diagnostic that provides direct measurement of IP performance, including queue dynamics at or beyond the onset of congestion. It uses a transport style sliding window algorithm combined with either ping or traceroute to sustain packet queues in the network. It can directly measure such parameters as throughput, packet loss rates and queue size as functions of packet and window sizes. Other parameters, such as switching time per packet or per byte, can also be derived. Many of the measurements can be performed either in a test bed environment (yielding the most accurate results), on single routers in situ in the Internet, or along specific paths in the production Internet. We will illustrate several measurement techniques.

I. Rationale

The Internet has experienced continuous rapid growth, sustained by the integration of prototype technologies into its production infrastructure. This "rapid deployment" is perhaps the hallmark of the Internet, but is not without its costs. Often, unproven technologies have been inflicted upon users who depend on the Internet for their basic activities.

The mission of the NSF Supercomputing Centers is to deliver leading edge supercomputing resources and services to a large user community widely dispersed across Internet. The networking effort at the Centers has always been in support of this mission: to deliver the best possible service at the highest possible performance to users and supercomputering applications. Pure network research often plays a lesser role than the pragmatic difficulties of integrating the newest commercial products in a high performance production environment.

The Pittsburgh Supercomputing Center has always been a part of this process, polishing new products and technologies for use in support of high performance applications and access for our users. The PSC was among the first sites directly attached to each implementation of the NSFnet: 56 kb/s, T1 (1.5 Mb/s) and currently T3 (45 Mb/s). Soon a connection will be installed to the NSF vBNS at OC-3 (155 Mb/s). Throughout its history, the PSC has been an early user of new technology.

The need to understand the performance limitations of unproven IP equipment and technologies has driven the development of diagnostics which overcome some of the limitations of traditional IP performance tools.

Most prior IP performance tools have been rate based [1]. These diagnostics are unstable across the onset of congestion because infinitesimal changes in relative traffic rates or switching performance causes the queues in the network to toggle between full and empty. Generally, the tester has precise control of the packet arrival rate. If it is less than the forwarding rate for a switch, the utilization will be less than one and there will not be a sustained queue in the switch. If the test traffic is infinitesimally faster than the switch's sustainable rate, it will quickly develop a maximum sized queue. Since discarding packets incurs a cost that is almost always different than forwarding packets, the switch operates at some poorly defined point beyond simple congestion. Constant rate tools can measure performance at rates below queue formation, but only at ill defined points beyond congestion. They can not measure switch performance under conditions of sustained partial queues.

Windowed ping addresses this problem. It uses a sliding window style flow control algorithm to keep a specific quantity of data outstanding in the network. As the window gets larger, a queue forms behind the bottlenecked component of the path. Windowed ping measures throughput (delivered data) and packet loss as functions of packet size and window size. The packets can either be ICMP echo (as in "ping") or low TTL UDP packets (as in "traceroute").

This approach can be adapted to any switching fabric that supports IP. The measured path can either be through an isolated piece of equipment in a test environment or across the production Internet.

Our goal is to raise the quality of IP implementations and the deployed infrastructure of the Internet by giving network providers and researchers the tools to properly document weaknesses in IP services and products. It is our hope that switch, router and network providers will evaluate their own products in anticipation of commentary from their customers.

II. The Algorithm

Windowed ping is based on a transport style sliding sequence window [2, p. 175] which is used to keep a specific quantity of data in transit in the network. Each transmitted packet and subsequent returning reply carries a sequence number that is used to control the transmission of new packets. Each reply causes new packets to be sent, so that a fixed number of packets is maintained in transit in the network.

Windowed ping uses two different types of transmitted packets. It can send ICMP echo requests [3], which cause the target to return a same length ICMP echo reply, as in the classic "ping" program [4]. Windowed ping can also send low time–to–live (TTL) UDP packets in the same manner as the "traceroute" diagnostic [5]. These generate minimum sized (48 byte) advisory ICMP messages as replies. The messages are typically "TTL exceeded" but others are possible, including "port," "host," or "network unreachable" messages. The ICMP messages include the transcribed header of the original packet and reflect its sequence number.

During each sample interval (typically 1 second), the parameters, packet size, window size (number of packets in the network), and TTL (if sending UDP packets) are held constant while the window algorithm maintains the specified number of packets in flight. If the network is not losing or reordering packets, each returning message causes exactly one new packet to be sent. If the network is losing packets, some of the returning messages will cause two or more packets to be sent, so that the sequence number of last packet sent is always greater than the last packet received by the currently selected window size. At the end of the sample interval, the number of packets transmitted and received is reported, and out–of–order and other exceptional packets are flagged.

The control parameters, TTL, packet size, and window size, can be held constant or automatically varied to scan the characteristic of a particular network path. The output is normally a table of sample intervals, listing the parameters and packet counts, and at the end, the grand total of received and transmitted packets. It is designed to be easily parsed for later data analysis.

III. Examples

The examples presented here are only to illustrate the behavior of windowed ping. They expressly do not represent benchmarks of any particular products. We anticipate that other researchers will perform rigorous and detailed benchmarks.

III.A. A simple measurement

Figure 1 represents a simple measurement of the IP performance of a low–end workstation. The path from the tester to the workstation was 5 hops through PSC's normal production network infrastructure. The workstation performance was measured by sending 1500 byte UDP packets with a TTL of 5 and varying the number of packets in flight.

The data reveals several charteristics of the workstation. In this configuration, it can only process about 490 packets per second (about 6 megabits per second.) Although this follows a different code path than TCP, it is likely to indicate a bound on TCP performance, because TCP also uses the socket interface between kernel and user mode. The flat portion of the data (ignoring the notches for the moment) represents a stable queue within the workstation. During this interval, raising the window size does not significantly change the performance. It only makes the queue longer.

To the left of the first knee, the window size is not large enough to fill the network. The throughput is dominated by the path delay which can be calculated from the graph. At any given window size, the path round trip time is the ratio of the number of packets in flight to the number of packets transmitted in one second. This particular path is about 10 ms under light load but rises to 20 ms near the knee. Usually the knee is

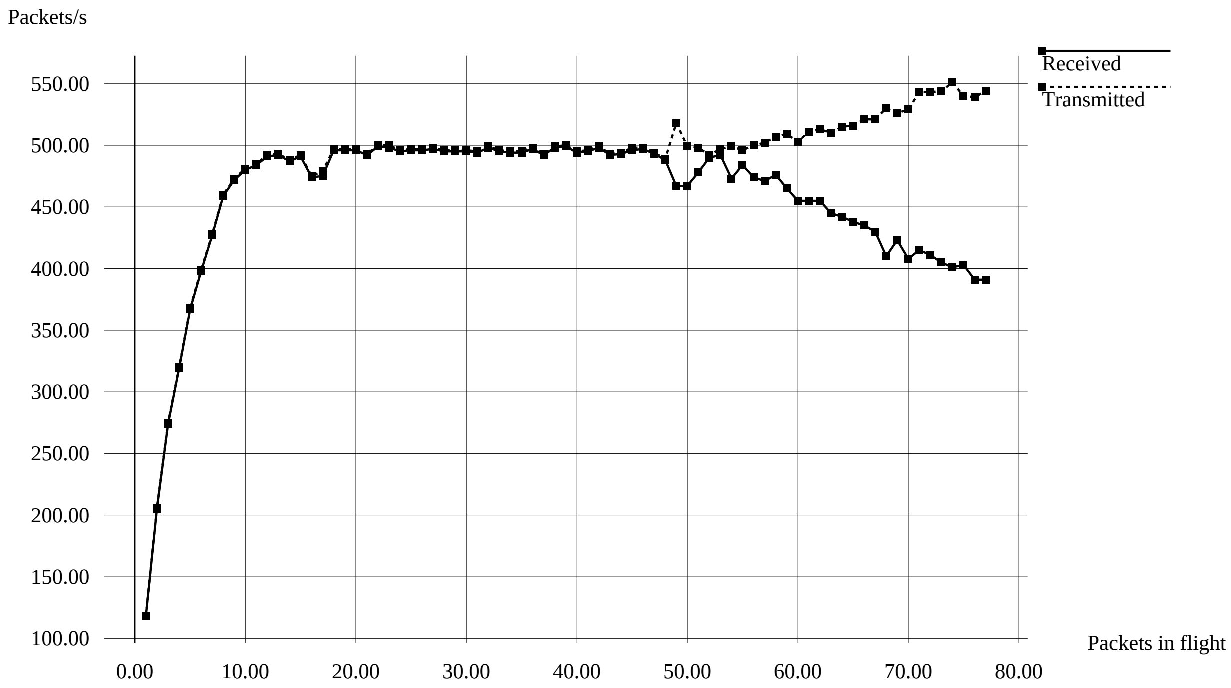

Figure 1: A Simple IP Performance Measurement

very sharp, as we shall see in other traces.

When there are more than 52 packets in flight, the IP path within the workstation becomes congested. It receives packets faster than it can process them and can send replies for only some of them. The slopes beyond 52 packets in flight give some clues to the loss mechanism. On this workstation, the bottlenecked resource is common to both the transmit and receive sides of the network interface. The relative slopes of the transmit and receive rates indicate that it is about twice as expensive to receive one 1500 byte packet as to send one 48 byte ICMP reply.

The first step in determining queue lengths is to estimate the locations of the knees. A fairly reliable method is to use the intersection of the asymptotes, which would put the knees at 10 and 52 packets in flight. This indicates that a queue starts to form when there are 10 packets in flight and starts to overflow when there are more than 52 packet in flight. Therefore the queue can hold 42 packets. A more accurate estimation requires a better understanding of the mechanism causing the curvature at the first knee.

The notches at 16 and 50 packets in flight are caused by routing updates. The update at 16 packets caused only 3 lost packets which are not visible in figure 1, but the update at 50 pushed the workstation into congestion, causing roughly 100 packets to be lost.

III.B. The NSFnet

Figure 2 shows a measurement of the production NSFnet [6] between PSC and San Diego Supercomputer Center in southern California. All seven hops are IBM RS/6000 workstations running AIX, with "smart" RS/960 cards in the FDDI and T3 interfaces. The RS/960 interface cards contain a complete microcode implementation of the IP protocol, including IP options, the ICMP protocol, and huge routing tables. The data was collected early in the morning to minimize the effect of other traffic on the measurement and the disruption to people who might be trying to use the NSFnet.

The path was measured with 4136 byte packets, which is a plausible size for TCP with MTU discovery (4kB payload + headers). Hop 1 is the FDDI between the tester and the edge of the NSFnet at PSC. Most of the data for this hop is off scale. Hops 2 and 3 aggregate traffic from several sites in the Ohio area onto the transit links of the backbone itself. Hops 4, 5 and 6 traverse west and south to Los Angeles, California. Hop

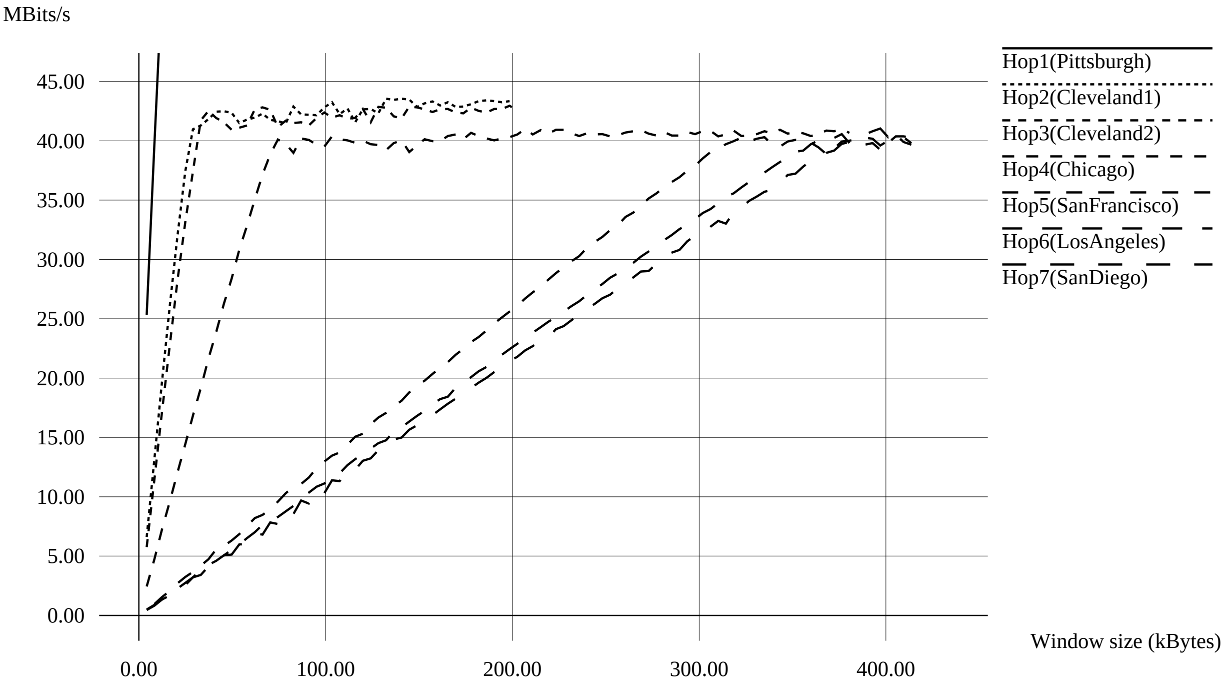

Figure 2: NSFnet IP Throughput, PSC to SDSC

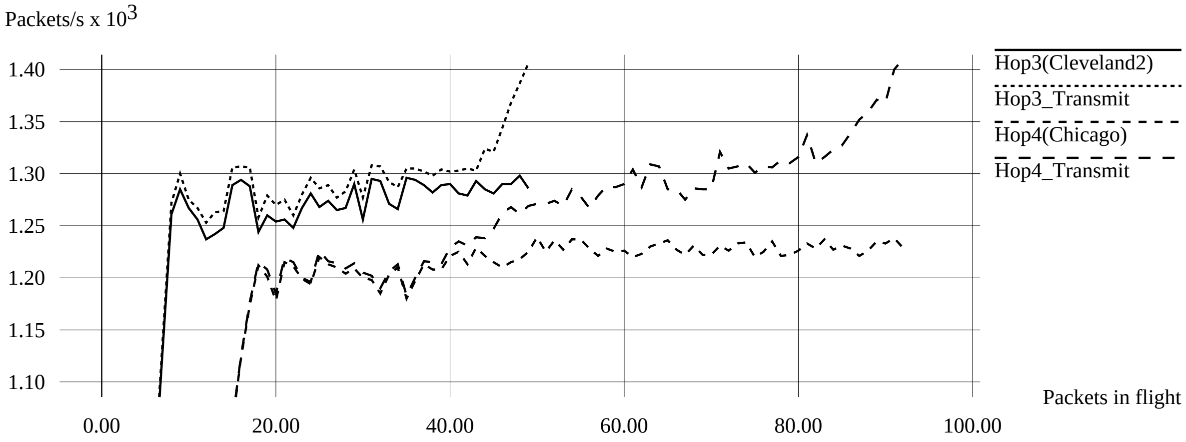

Figure 3: NSFnet IP Throughput and Transmit Rates, Hops 3 and 4 to SDSC

7 is the edge of the NSFnet at SDSC. The entire path is about 4000 kilometers. Table 1 lists the delays to each hop, computed from the leading faces. These delays agree with the delays as reported by the direct measurement in traceroute.

Delay from PSC to each router in figure 2.		
Hop	Delay	City
1	1 ms	Pittsburgh PA
2	5 ms	Cleveland OH
3	5 ms	Cleveland OH
4	13 ms	Chicago IL
5	61 ms	San Francisco CA
6	70 ms	Los Angeles CA
7	74 ms	San Diego CA

Table 1

While figure 2 is not suitable for estimating queue sizes, it does reveal several other interesting artifacts. The bandwidth fluctuations near queue formation (between 30 and 100 kByte window) are present in every lightly loaded test of the NSFnet. They are probably artifacts of "batching" in the device drivers. Small groups of packets are processed together, saving some of the interrupt overhead. The precise efficiency gain depends on the relative timing of the batches in one direction and the replies in the other.

The path to hop 3 (Cleveland2) is about 10% faster than the path to hop 4 (Chicago). A possible explanation for this difference is that the line to Chicago is fairly busy, even early in the morning. Other traffic may indeed depress the measurements for hop 4. However, other traffic tends to create a distinctive upward tilt in the flat part of the trace because the sustained queue causes other more polite protocols (e.g. TCP) to detect congestion and appropriately reduce their rates.

Figure 3 provides a more detailed look at hops 3 and 4, including throughput and transmit rates. The knees for hop 3 are at 8 and 43 packets in flight so the queue length is 35 packets. The shape of the graph shows throughput and transmit rates at queue overflow typical of a saturated serial link. There is a phenomenaly uniform 20 pps loss, starting as soon as a queue forms (at 8 packets in flight). It is possible that that this represents a formal gateway congestion control measure [7] such as random drop [8] or Random Early Detection [9]. Since this loss is only present in the path to hop 3, it is more likely to be an artifact of some problem that only manifests itself when there is a sustained queue. The prime suspect is the output queue at hop 2, which is feeding the link between the two routers in Cleveland.

The data for hop 4 is significantly more complicated, reflecting the effect of cascaded routers with slightly different bandwidths. Between 18 and 40 packets in flight, a queue accumulates in Cleveland2 (hop 3) behind the saturated link to Chicago (hop 4, at about 1200 packets per second). Beyond 40 packets in flight, this queue becomes full and overflow packets are discarded (at hop 3). However, the path to hop 3 is limited to 1300 packets per second. Between 40 and 80 packets in flight, a queue forms at hop 2 while hop 3 continues to discard packets at a rate equal to the difference between its input and output links, roughly 90 packets per second. In this region, the throughput is 1210 packets per second while the transmit rate approaches 1300 packets per second. Beyond 80 packets in flight, the queue at hop 2 also fills, and the total loss rate rises steeply. The data here is noisy enough to make it difficult to estimate queue sizes.

It takes about 90 packets (375 kBytes) to fill the entire path from PSC to SDSC.

III.C. A DEC Alpha as a router

A second tool, *slack,* measures the idle time of a computer system by setting its priority as low as possible and counting the number of *gettimeofday* system calls it can execute in each second. In effect, *slack* replaces the system idle loop. It is extremely sensitive to system activity, including device interrupts. On typical workstations it easily detects small mouse motions, even when not posting X events. One problem with *slack* is that it measures slightly ill-defined resources. While it measures unused CPU cycles, it tends to reflect memory cache flushes and contention for memory bandwidth because these affect CPU performance. However, experience suggests that this is not a problem since measurements with *slack* are highly repeatable and smooth functions of traffic rates.

Figures 4, 5, 6 and 7 show measurements of the switching performance of a Digital Equipment Corporation AXP 3000/400 workstation ("Alpha") configured as an FDDI to FDDI router. The test was between two Silicon Graphics Indigo workstations, using large UDP packets in one direction and tiny ICMP packets in return.

The Alpha acts as a router and only exercises its IP path. The test represents a plausible network workload because TCP typically sends large

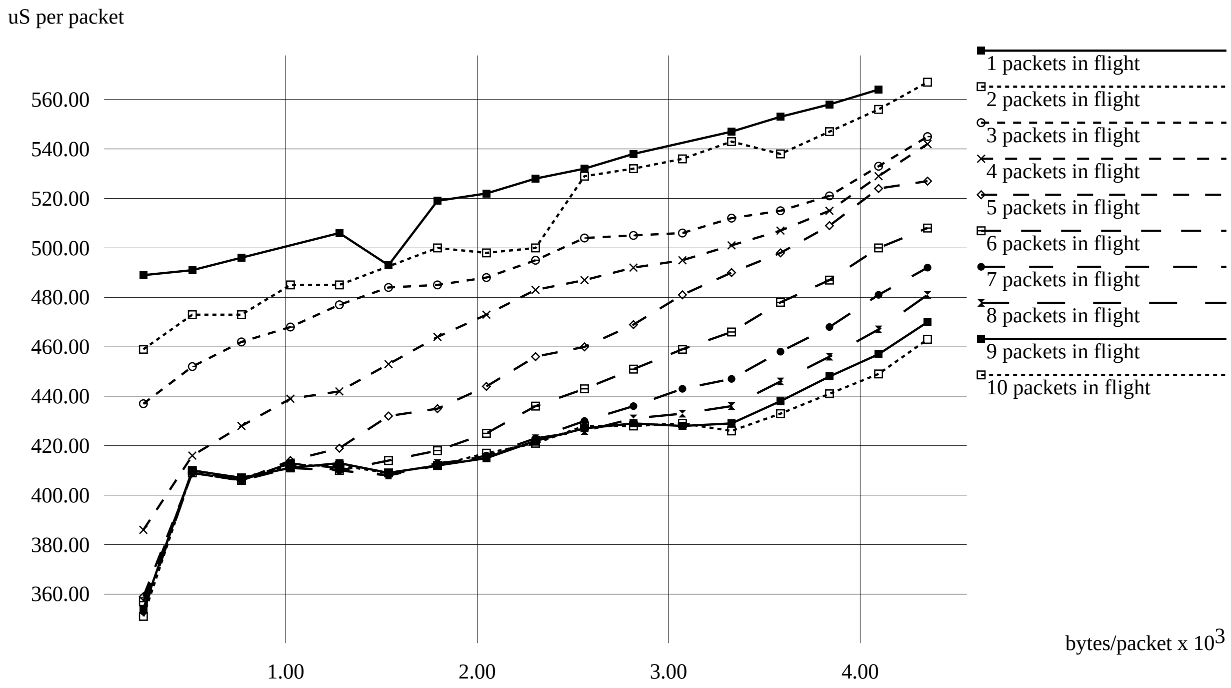

Figure 7: DEC Alpha, Packet Switching Time

data packets in one direction while sending tiny ACKs in the opposite. The statistics are not the same as a real workload, but this is not necessary for measurements with *slack*.

All four figures are different views of the same data, collected with windowed ping and *slack*. Of the nearly 200 sample intervals, only 6 were discarded (due to other activity interfering with the measurement). The complete run took about half an hour and processed almost 3 million packets. The total loss was 691 packets, or slightly more than 0.02%.

Figure 4 is in the same form as the previous examples. It is not particularly useful in this test because tester artifacts can not easily be distinguished from router behavior. The round trip time is only about 1 ms, which includes the tester, the "reflector," and two passes through the Alpha under test.

Figures 5 and 6 show the available *slack* time, normalized as percentage of the quiescent idle time. In figure 5, it is plotted as a function of throughput and packet size; in figure 6, as a function of packets rate and packet size. The close clustering in figure 6 suggests that routing performance of the Alpha is dominated by cost per packet and not per byte.

Figure 7 shows the measured switching time as a function of packet size. These times were computed from the *slack* data by allocating the missing *slack* time across the packets processed. To facilitate analysis, CPU time is partitioned into four distinct activities: copying the data, processing data packets (excluding the data itself), processing the replies, and *slack*. Other activities are insignificant or uniform across all relevant samples. Any sample interval which included other activity, such as system daemons, was discarded.

One phenomenon that makes interpreting the data difficult is that there is a significant difference in performance between switching isolated and back–to–back packets. There are at least two possible explanations for this. The memory system includes cacheing, which raises the performance of re–executed code. It is also possible that there is batching in the device drivers. It would be necessary to conduct additional experiments or have better information about the implementation to completely understand this phenomenon.

Assuming that the time to forward each small data packet (256 bytes) is the same as the time to forward the replies, the left most data points

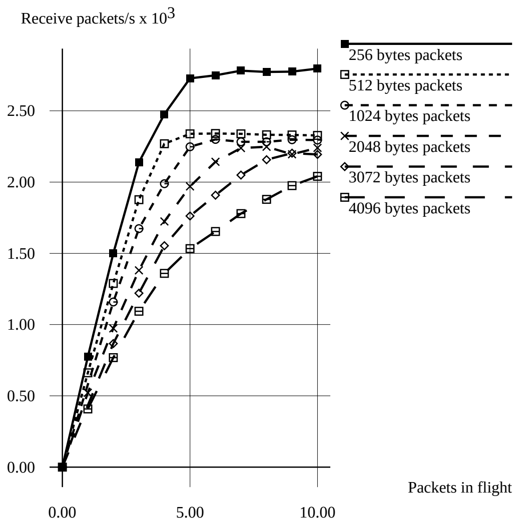

Figure 4: DEC Alpha, Path Performance

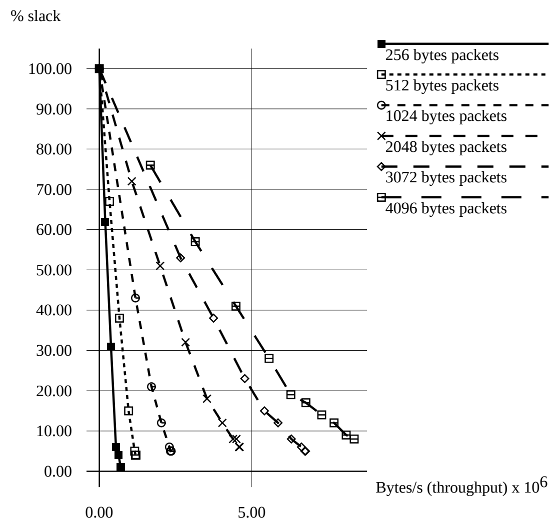

Figure 5: DEC Alpha, Idle Time

show the switching time for each packet plus the time for each ACK. Thus the times are for two switched packets. Each forwarded packet takes between 175 μs and 245 μs depending on its headway.

The large jump between 256 byte and 512 byte packets for the large window cases is due to data packets spilling over into a second kernel memory buffer. For some reason, this does not affect the performance of isolated packets. For closely spaced packets, the second memory buffer costs 60 μs.

The slope of the upper and lower edges of the envelope is 20 ns/byte which closely agrees with the known performance of the memory system. (80 ns per 8 byte word, with 2 accesses per each word forwarded).

In this example, windowed ping and *slack* have been used to estimate the performance of a workstation used as a router. A detailed analysis of the traces within the envelope requires a more complete description of the system internals.

III.D. A commercial router

Figures 8 and 9 illustrate the performance of a distant commercial router, measured from Pittsburgh via the NSFnet. The first 5 hops of the path are the same as figure 2 and table 1. Hop 6 aggregates traffic in the San Francisco area onto the NSFnet. Figure 8 shows the performance of the path to hop 7, the ENSS at the edge of the NSFnet, and to hop 9, a workstation beyond the router under test (hop 8). The round trip times to hops 7 and 9 are 61 and 63 ms respectively.

Although the entire path supports 4352 byte packets, this measurement was performed with 1500 byte packets, as though the final destination were an ethernet connected workstation.

The downward slope beyond 140 packets in flight is typical of a queue that has increasing overhead as it gets longer. For hop 7, other experiments reveal that the queue in question is the reply input queue to the windowed ping process on the tester. This is a standard socket buffer, which is implemented with a singly linked list. The kernel inserts data into the socket by searching through the linked list. Normally the tester is not the bottleneck and this queue is always nearly empty. If the tester can not keep up with its own replies, the queue grows and causes it to slow down further. The downward slope at hop 9 is due to some other queue with increasing overhead, probably associated with the workstation at hop 9. If this were a router, decreasing performance as a consequence of increasing load would be viewed a serious flaw.

Other sources [11] suggest that the main line IP code path in the router at hop 8 can not be stressed in this configuration. In this router, the main line IP path is implemented in special hardware and microcode, while other less used parts of the protocol are implemented on another processor entirely. Figure 9 shows the performance of the router when exercising this alternate path

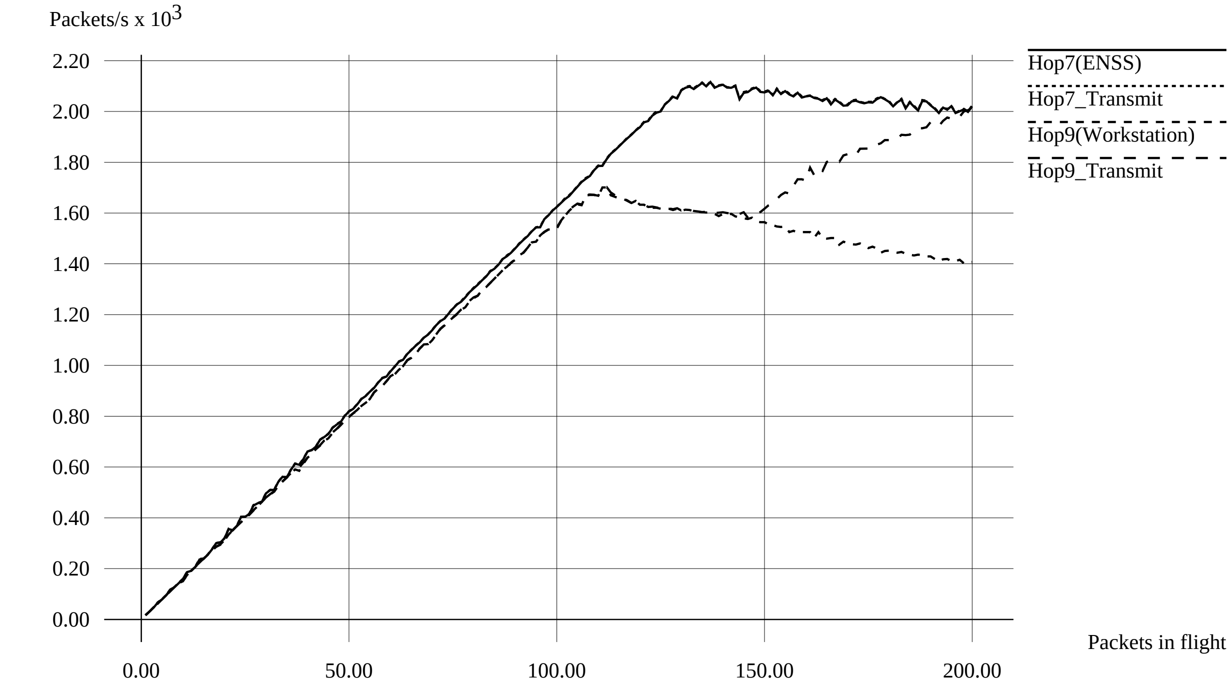

Figure 8: Router Performance via the NSFnet, Hops 7 and 9

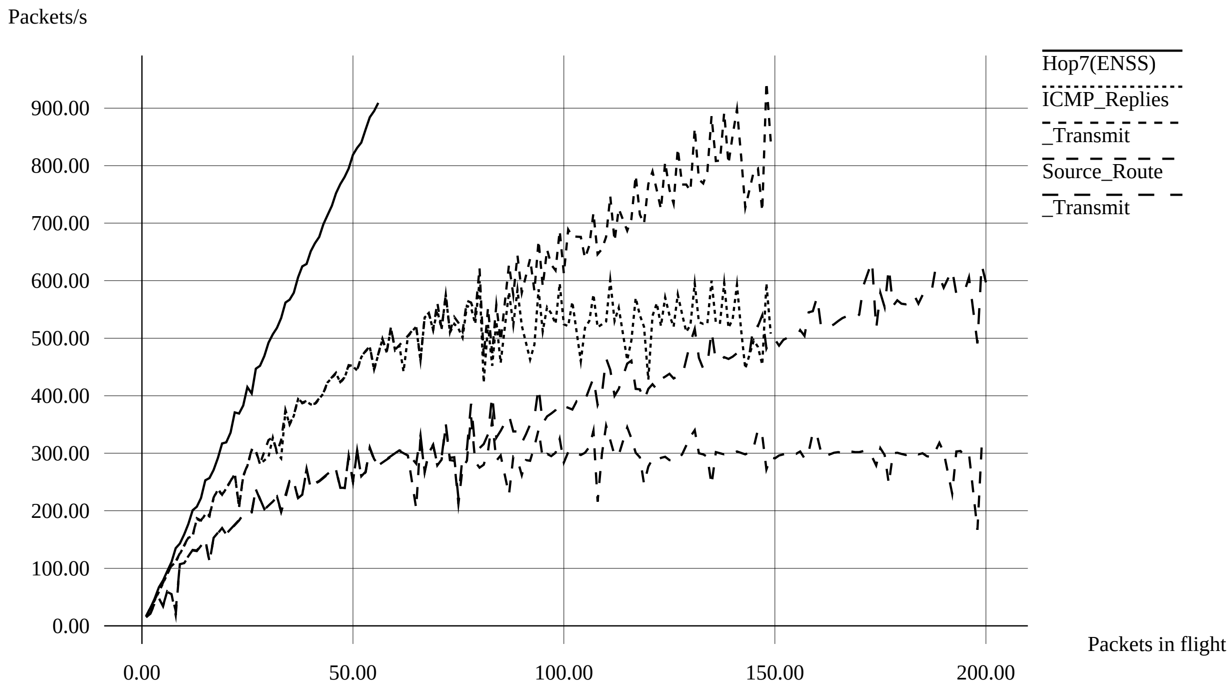

Figure 9: Alternate Path Performance for the Router at Hop 8

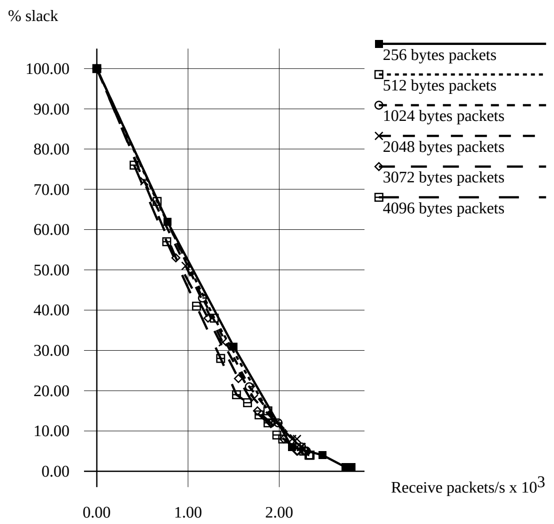

Figure 6: DEC Alpha, Idle Time

by measuring its ability to return ICMP messages and to process source routes (the IP option "LSRR")[12]. Both of these are mandatory parts of the IP protocol. For comparison purposes, the path characteristics to the ENSS are shown.

All three traces were collected in one run of windowed ping. The packets all contain an IP source route via the workstation, and an UDP payload, such that the total packet size is 1500 bytes. The path parameters are measured at TTLs of 7 (to the ENSS), 8 (to the router) and 9 (to the workstation). It is important to remember that in each case, the measured performance is that of the entire path to the given hop.

The difference in performance of the path to the ENSS (hop 7) with and without the source route (figures 8 and 9) is not significant. In figure 9, the leading edge is shown for comparison with the other two traces, but the upper portion is omitted because it is off scale.

The router can only generate about 550 ICMP messages per second. The delay at the point where loss begins (75 packets in flight) is 140 ms. The shape of the graph where the transmit rate continues to rise beyond the point where the throughput reaches a plateau is typical of a router with no significant queue. It is as though the knees for the start of queue formation and queue overflow occur at the same point. This may be appropriate behavior for generating ICMP messages, since no applications except windowed ping depend on the network returning every single ICMP message. This test is not sufficient to determine whether or not the presence of the source route option affects the performance for generating the ICMP message.

The source–routed packets are processed at only 300 packets per second (3.6 Mbit/s). Even though this is a state of the art FDDI to FDDI router, it can not even fill half of an ethernet with source–routed packets. The round trip time at the start of queue formation (about 60 packets in flight) is 200 ms. The delay within the router is more than twice the delay through the rest of the path. There does appear to be a slight offset between the upper and lower sections of the transmit trace, suggesting that the knees for queue formation and queue overflow are at different points, and that there is a queue for the bottlenecked resource within the router. However, the queue is not significant relative to other fluctuations in performance. The additional delay indicates that there must be at least 50 packets within the router, probably distributed across more than one queue.

Note that in this case the ICMP messages being used by windowed ping are returned by the workstation at hop 9, which is not the performance bottleneck in the path. This same technique can be used to measure the performance of other non-IP components such as bridges and ATM switches.

IV. Conclusion

Windowed ping has greatly enhanced PSC's ability to quickly test and evaluate the IP performance of new products and technologies. It has been instrumental in our ability to deploy leading edge networking components in our production facilities.

Simple measurements, as shown in figure 1, require no more setup than already necessary to attain basic management access to a router. Most experiments take no longer than a few minutes, and yield a simple, accurate picture of the IP performance. Other measurements take longer to set up and perform, but yield correspondingly richer results.

We are making this software available free of charge to vendors, researchers and network providers. The package includes *slack* and the data reduction software used in this paper. For more information, please send a request to wping@psc.edu.

Acknowledgements

I wish to thank the following for their contributions to this work: Jamshid Mahdavi for examples and his extensive field experience with windowed ping, everyone at PSC for their support, National Science Foundation for funding the network test platforms (NCR-9314768), and my participation under the PSC cooperative agreement (ASC-8902826).

I would also like to thank the people who run the NSFnet, at Merit and Advanced Network & Services, for tolerating my testing. I apologize to anyone who suffered as a result of it.

I wish to thank Vince Fuller and the people at BARRNet and Stanford for providing an interesting router to test.

And finally, I would like to thank my wife, Susan Blackman, who contributed greatly to the clarity of this paper.

References

[1] Work in Progress, Benchmarking Methodology Working Group of the Internet Engineering Task Force.

[2] D. Comer, "Internetworking with TCP/IP, Volume I," Prentice Hall International, 1991.

[3] J. Postel, "Internet Control Message Protocol," RFC792, USC/Information Sciences Institute, September 1981.

[4] M. Miller, "Troublshooting TCP/IP," M&T Books, 1992.

[5] V. Jacobson, "traceroute," Documentation and source available via ftp from ftp.ee.lbl.gov.

[6] B. Chinoy, H.-W. Braun, "The National Science Foundation Network," San Diego Supercomputing Center technical report GA-A21029, 1992.

[7] V. Jacobson, "Congestion Avoidance and Control," Proc. ACM SIGCOMM '88, August 1988, pp. 314-329.

[8] A. Mankin, "Random Drop Congestion Control," Proceedings of SIGCOMM 90, September 1990.

[9] S. Floyd, V. Jacobsen, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, August 1993.

[10] S. Bradner, "Ethernet Bridges and Routers," Data Communications, Feburary 1992.

[11] S. Bradner, "Ethernet Bridges and Routers," Presention at Interop in August 1993, (no proceedings). An electronic version is available via gopher://ndtl-gopher.harvard.edu/11/ndtl/results.

[12] J. Postel, "Internet Protocol," RFC791, USC/Information Sciences Institute, September 1981.

Author Information

Matthew Mathis is a network engineering specialist at the Pittsburgh Supercomputing Center, responsible for the design and implementation of high performance data communications. His work includes development of high performance networking tools and diagnostics, tuning of high performance networks connecting PSC's supercomputers, and research into network performance issues on both local area and wide area networks. Mathis was a key developer of the Border Gateway Protocol, used to route IP traffic over the global Internet. He was instrumental in porting the Andrew File System to Cray Unicos and implemented the first distributed application using a HiPPI channel to share data between heterogeneous supercomputers (a Cray YMP and a Thinking Machines CM-2). Mathis received an M.S. in Electrical Engineering from Carnegie Mellon University.

[This draft differs from the final version to be presented at INET '94]