Packet Reordering, High Speed Networks and Transport Protocol Performance

Ladan Gharai
University of Southern California
Information Sciences Institute
Email: ladan@isi.edu

Colin Perkins
University of Glasgow
Department of Computing Science
Email: csp@csperkins.org

Tom Lehman
University of Southern California
Information Sciences Institute
Email: tlehman@isi.edu

***Abstract*— We performed end-to-end measurements of UDP/IP flows across an Internet backbone network. Using this data, we characterized the packet reordering processes seen in the network. Our results demonstrate the high prevalence of packet reordering relative to packet loss, and show a strong correlation between packet rate and reordering on the network we studied. We conclude that, given the increased parallelism in modern networks and the demands of high performance applications, new application and protocol designs should treat packet reordering on an equal footing to packet loss, and must be robust and resilient to both in order to achieve high performance.**

I. Introduction

The presence of packet reordering in IP networks has long been known. Until recently though, it was thought of as pathological behavior rather than a normal part of the network's dynamics [2]. Accordingly, many protocols and applications have been designed assuming in-order packet delivery or, at best, with a very low tolerance to out-of-order delivery. For example, transport protocols such as TCP routinely treat certain instances of out of order packet delivery as a congestion signal.

It is now recognized that this behavior is not ideal, and that transport protocols should be designed to be tolerant of packet reordering. Despite this, there has been little research undertaken to determine when reordering is likely to occur. This makes it difficult to decide the relative importance of tolerance to packet reordering compared with tolerance to other network events.

In this paper we study the occurrence of packet reordering on a commercial IP backbone network, reporting on the variation in reordering rate dependent on the packet size and inter-arrival times. We discuss the likely future development of high speed network infrastructure, and explore how these developments will affect the rate of packet reordering, and hence transport protocol performance.

The paper is organized as follows: in Section II we outline our motivation in conducting these experiments and describe, in more detail, the problem we address. In Section III, we describe our experimental methodology, followed by the empirical results in Section IV. Section V discusses the implications of these results for application and protocol designers. Finally, we conclude in Section VI.

II. Motivation

Previously, we have developed a prototype teleconferencing system that uses High-Definition TV (HDTV) equipment to deliver very high quality video over IP networks [15]. The system uses RTP over UDP/IP as its network transport [9, 16] and contains a preliminary TCP friendly rate control (TFRC) implementation [7, 10] to adapt its native data rate, approximately one gigabit per second, to the network capacity.

While performing experiments to test the throughput and congestion response of this system we were confronted with an interesting phenomenon: the system was sending at a lower rate than expected, given the degree of packet loss observed. Further investigation [8] determined this was due to packet reordering within the network, with the rate control algorithm inferring spurious congestion events due to the presence of significantly out of order packets. These spurious congestion events appeared to occur most often when the system was transmitting at high rate, and caused a precipitous decline in sending rate.

Such a response to packet reordering is dictated by the TFRC congestion control mechanism, which ensures fairness and compatibility with TCP flows by mimicking TCP's interpretation of certain patterns of reordering as a congestion signal. While the merits of this reaction to reordering for multimedia applications are certainly questionable, and provide an interesting area of research, the point that caught our attention was the possible existence of a correlation between packet reordering and data rate.

This leads us to the following question: does an increase in data rate, which is achieved by maintaining a constant packet size and increasing the packet rate, thereby reducing packet inter-arrival time, result in an increased rate of occurrence of packet reordering? And, if so, what are the implications of this in the design of network transport protocols?

III. Methodology

The aim of our experiments is to observe and characterize the sequenced behavior of end-to-end flows across an IP backbone network as a means to evaluate the effects of packet reordering on multimedia and TCP/IP based applications. To this end we: (1) setup a measurement testbed; (2) generated UDP flows with a range of data rates and different packet sizes over three different paths on the testbed; and (3) evaluated each

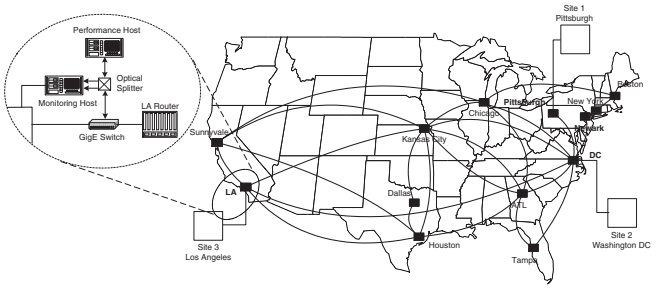


Fig. 1. Testbed configuration

flow for occurrences of loss and reordering. In the following, we describe the components of our experimental methodology in more detail.

A. Measurement Testbed

The configuration of our testbed is shown in Figure 1. We installed traffic generation and monitoring systems at three sites – the USC/ISI-East laboratory in the Washington DC metro area, CMU in Pittsburgh and the network operations center at One Wilshire Boulevard in Los Angeles. These sites were the peering points for three DARPA SuperNet partners [5], chosen because they maintained direct peerings with the ISP at OC-48 rate, and because we were allowed to connect our equipment directly behind the access router, hence eliminating the effects of the edge network.

For the traffic generation systems, we used PCs with dual 1.8 GHz Xeon processors running Linux 2.4.19, equipped with SysKonnect SK-9843 gigabit Ethernet cards. The monitoring hosts were uniprocessor 1.8 GHz Xeon PCs, with 4 gigabytes of memory, UDMA-100 disks, dual SysKonnect SK-9843 gigabit Ethernet cards and a fast Ethernet interface, running FreeBSD 4.5. These were chosen after local tests demonstrated that the traffic generation hosts could saturate the gigabit Ethernet with traffic, and could sustain this without packet loss or reordering with packet sizes of at least 1500 octets (with a packet size of 500 octets, rates up to 500 Mbps could be sustained), and that the monitoring hosts could record packet headers at matching rates.

The traffic generation hosts were connected to the core router at each site via a single gigabit Ethernet switch. An optical splitter was used on the fiber connecting the traffic generation host to the switch, directing a copy of all outgoing traffic to the receive port of the first gigabit interface on the monitoring host, and a copy of all incoming traffic to the receive port of the second gigabit interface on the monitoring host. Since the transmit lines of the gigabit Ethernet ports on the monitoring host are not connected, this provides a non-intrusive optical network tap, used to capture all traffic on the link. The switch ensures that only traffic sourced by, or destined for, the traffic generation host is seen by the monitor, ensuring the privacy of other users of the network.

The monitoring hosts were based on the Self Configuring Network Monitoring [1] systems developed by Lawrence Berkeley National Laboratory Data Intensive Distributed Computing Research group. These machines run a modified version of FreeBSD 4.5 that allows bonding of two network interfaces such that a single Berkeley packet filter can receive packets from multiple interfaces. Other kernel and driver modifications included changes to reduce interrupt load and provide more accurate timestamps. The result was a system which could listen to the receive ports of two gigabit Ethernet network interface cards, and capture all packet headers along with a small amount of payload data at gigabit rates. Data is captured using a version of `tcpdump` modified to limit disk access by buffering data in memory until each test is complete. Each traffic generation host has an associated monitoring host, enabling us to record complete packet traces of all IP traffic at the sender and receiver.

At each site, the role of the traffic generation system was to actively introduce UDP/IP flows into the network, while the monitoring hosts at the source and destination recorded the traffic for later analysis. In addition, we logged the route taken, values of tuning parameters for the host network stack, and throughput and packet loss rates.

B. Test Flows

We gathered pairwise measurements of UDP/IP packet flows, generated with `iperf v1.1.1` [18], between all three sites. Test flows were one minute duration, at rates of 1 Mbps, 10 Mbps, 100 Mbps and at 100 Mbps intervals up to and including 900 Mbps. We repeated each test with packet sizes of 500, 1500 and 4500 octets.

To describe a particular test flow, we henceforth use the following notation: $F_b(s \rightarrow d, m)$. Where s and d refer to the source and destination of the flow, and m indicates the packet size. Each flow is also subscripted by the rate at which they were generated. For example, to describe a 300 Mbps flow between from DC to LA with an MTU of 1500, we use the notation: $F_{300}(\text{DC}{\rightarrow}\text{LA, } 1500)$.

We conducted our analysis of packet reordering using UDP flows because this allows us to probe the network in a controlled manner. The sending rate of a UDP flow, unlike that of a TCP flow, is entirely under application control allowing us to generate UDP flows with fixed known packet size and spacing, whereas the transmission rate of a TCP flow varies according to a complex congestion control algorithm. This lets us conduct controlled experiments to determine the influence of packet size and inter-arrival times on packet reordering. Clearly controlling packet spacing and data rate is not possible with TCP, as a TCP sender's prime concern is maintaining fairness with other flows, while seeking available bandwidth.

C. Metrics

Each flow is analyzed for both packet loss and reordering (since we conduct offline analysis of complete packet traces, we are able to distinguish loss from reordering in all cases). We evaluate packet reordering according to two different metrics, one based on monotonic increase of sequence numbers, the other more TCP-like.

Monotonic Increase: An often used metric of reordering is the notion of monotonic increase of packet sequence numbers: provided the sequence numbers increase in a continuous and monotonic sequence, packets are deemed to be in order. Otherwise, all packets are deemed out of order, until a packet with sequence number greater than the last recorded in-order packet arrives. The monotonic increase metric, O_1, denotes a packet as out of order if Equation 1 is satisfied:

$$\begin{aligned} &\forall i,j,k : i < j < k \text{ and } S_j > S_i \text{ and } S_k < S_j \\ &\Rightarrow P_k \text{ is out of order} \end{aligned} \quad (1)$$

Where i, j and k index the arrival timeline of packets P_i, P_j and P_k at the receiver (such that P_i arrives prior P_j and P_j arrives prior to P_k). Previous studies of packet reordering [11, 14] have utilized monotonic increase as a means of quantifying orderliness. Accordingly, this metric forms a basis for comparison with some previously published data.

TCP-Like: The TCP-like packet reordering metric, O_2, counts reordering events that would cause a TCP connection to see a spurious congestion signal. A TCP connection uses an acknowledgment mechanism to report the highest sequence number received, and generates a duplicate ACK if packets arrive out of order. A triple-duplicate ACKs is interpreted as a congestion signal. The TCP-like metric, O_2, therefore counts reordering events that would cause a triple-duplicate ACK to be generated. These events are defined to occur when Equation 2 is satisfied:

$$\begin{aligned} &\forall i : (S_i < S_{i-1}) \text{ and } (S_i < S_{i-2}) \text{ and } (S_i < S_{i-3}) \\ &\Rightarrow P_i \text{ is out of order} \end{aligned} \quad (2)$$

For the monotonic metric, O_1, we measure and report the number of reordering events as a percentage of the total number of packets in a flow. For metric O_2, we list the total number of reordering events that occur during the duration of each flow, since TCP is affected by the number of congestion signals, rather than the ratio of congestion signals to total segments.

IV. Results

In Table I we list the results from the application of the reordering metrics O_1 and O_2 to UDP flows generated on the three paths in our testbed: Pittsburgh⇔LA, Pittsburgh⇔DC and LA⇔DC. Entries marked '-' reflect tests in which the monitoring hosts could not capture all the packets in the flow, as detected by gaps in the packet trace recorded at the source. A total of 155 complete UDP flows were analyzed, comprising approximately 60 million packets, over three different bidirectional network paths.

We used the `traceroute` utility to record the route traversed by each flow. It was verified that all flows on a particular route traversed the same links, and that the forward and reverse paths differed by at most one hop. Despite this, measurements of the forward and reverse paths exhibit asymmetrical properties. This effect has been noted by other researchers, and could be due to the effects of cross traffic.

We observe that packet loss rates are negligible, as expected on a modern IP backbone network. Packet loss occurred in only two flows: one packet was lost in F_{600}(DC→ Pitt, 1500), and 21 packets lost in F_{400}(Pitt→ DC,500). In total, 22 packets were lost in the network, from a total of approximately 60 million sent. The absence of packet loss leads us to believe that capacity is available, and that a TCP flow should be able to sustain high throughput in this environment.

A. Prevalence of Reordering

Of the 155 flows we analyzed, 73 flows (47%) contained at least one out of order packet. Of those, 48 flows saw more than 0.01% of packets reordered according to metric O_1. The largest amount of reordering we observed was 1.65% according to this metric, in F_{500}(Pitt→LA, 500) and F_{400}(Pitt→LA, 500). These results compare well with other recent studies of reordering in IP backbone networks. For example Jaiswal *et al* [11] report reordering of 0.02% to 0.5% of packets.

For a given packet size, M, and sending rate, b, the degree of reordering measured according to metric O_1 is relatively consistent on the different paths, F_b(src→dst, M). The exception to this is the DC→LA path, which also experiences disproportionally high values of reordering measured by metric O_2. While we can speculate as to why the DC→LA path experiences higher values of reordering (i.e., an awry router) unfortunately at this point we cannot further diagnose this path.

In terms of consistency between values of metric O_1 in Table I (with the exception of the DC→LA path), we observe that a majority of flows with 500 octet packets experience reordering at data rates of 200 Mbps and higher, while flows with 1500 octet packets experience reordering at data rates of 600 Mbps and above. There appears to be a clear threshold, coinciding with a difference in inter-packet arrival times of approximately 0.02ms, beyond which packet reordering will occur increasingly often. None of the flows with 4500 octet packets, which have inter-packet arrival times greater than 0.04ms at the data rates we tested, experience reordering.

B. Packet reordering and inter-arrival time

To investigate the relationship between packet timing and the likelihood of reordering, we compared the inter-packet delay at the source with the corresponding inter-packet arrival times at the destination.

Flows were initially generated with constant inter-packet timing, depending on packet size and required data rate. The inter-packet arrival time, as measured at the receiver, varies depending on the effects of the network. In Figure 2, we show two examples of the timing variation that is induced by the network: F_{800}(Pitt→ DC, 1500) and F_{300}(LA→ Pitt, 500). For packets delivered in-order, the figure shows the expected dispersion in inter-packet timing due to effects of jitter (for example, cross-traffic and queuing in routers). Packets delivered out-of-order, however, are more likely to arrive immediately after a previously delivered packet. We consistently noted this pattern of smaller inter-arrival times

TABLE I

SUMMARY OF REORDERING METRICS.

Path	Size	Metric	Rate (Mbps) 1	10	100	200	300	400	500	600	700	800	900
LA→DC	500	O_1%	0	0	0	0.07	0.45	1.26					
		O_2 (events)	0	0	0	0	0	0	-	-	-	-	-
	1500	O_1%	0	0	0	0	0	0	0	0.01	0.02	0.05	
		O_2 (events)	0	0	0	0	0	0	0	0	0	0	-
	4500	O_1%	0	0	0	0	0	0	0	0	0	0	0
		O_2 (events)	0	0	0	0	0	0	0	0	0	0	0
DC→LA	500	O_1%	0	0	0.05	0.16	0.81						
		O_2 (events)	0	0	15	191	783	-	-	-	-	-	-
	1500	O_1%	0	0	0	0.01	0.03	0.06	0.12	0.38	0.55		
		O_2 (events)	0	0	0	0	2	21	88	3299	1049	-	-
	4500	O_1%	0	0	0	0	0	0	0	0.01	0.02	0.06	0.13
		O_2 (events)	0	0	0	0	0	0	0	0	0	4	12
Pitt→DC	500	O_1%	0	0	0	0.01	0.41	0.57					
		O_2 (events)	0	0	0	1	4	30	-	-	-	-	-
	1500	O_1%	0	0	0	0	0	0	0	0.01	0.01	0.02	0.03
		O_2 (events)	0	0	0	0	0	0	0	0	0	2	8
	4500	O_1%	0	0	0	0	0	0	0	0	0	0	0
		O_2 (events)	0	0	0	0	0	0	0	0	0	0	0
DC→Pitt	500	O_1%	0	0	0.02	0.04							
		O_2 (events)	0	0	1	22	-	-	-	-	-	-	-
	1500	O_1%	0	0	0	0	0	0	0.01	0.01	0.01		
		O_2 (events)	0	0	0	0	0	0	0	2	0	-	-
	4500	O_1%	0	0	0	0	0	0	0	0	0	0	0
		O_2 (events)	0	0	0	0	0	0	0	0	0	0	0
Pitt→LA	500	O_1%	0	0	0	0.02	0.87	1.28	1.65				
		O_2 (events)	0	0	0	8	41	67	122	-	-	-	-
	1500	O_1%	0	0	0	0	0	0	0	0.02	0.04	0.07	
		O_2 (events)	0	0	0	0	0	0	2	9	30	30	-
	4500	O_1%	0	0	0	0	0	0	0	0	0	0	0
		O_2 (events)	0	0	0	0	0	0	0	0	0	0	0
LA→Pitt	500	O_1%	0	0	0	0.09	0.59	1.65					
		O_2 (events)	0	0	0	0	0	3	-	-	-	-	-
	1500	O_1%	0	0	0	0	0	0	0	0.01	0.04		
		O_2 (events)	0	0	0	0	0	0	0	0	0	-	-
	4500	O_1%	0	0	0	0	0	0	0	0			
		O_2 (events)	0	0	0	0	0	0	0	0	-	-	-

for all out-of-order packet arrivals across all flows: it seems likely that parallelism at the link layer will allow some packets to "catch-up" with those sent earlier but queued on a parallel link, causing this behavior.

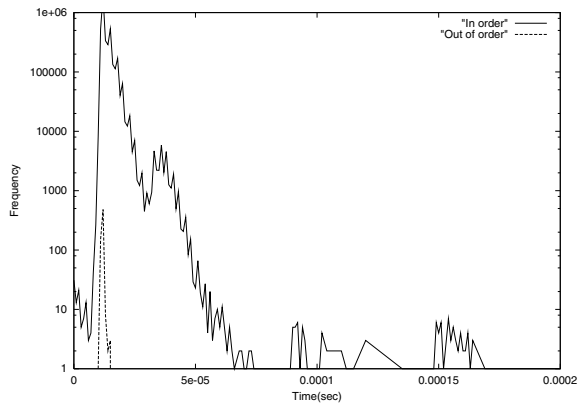
C. Packet Reordering and TCP

The data in Table I illustrates that the relationship between the fraction of reordered packets according to metric O_1 and the number of packet reordering events that would affect a TCP flow, metric O_2, is non-linear. It does not appear to be possible to predict the reaction of a TCP flow to packet reordering using a simple metric based on the fraction of packets reordered, unless that metric also includes the effects of the pattern of reordering.

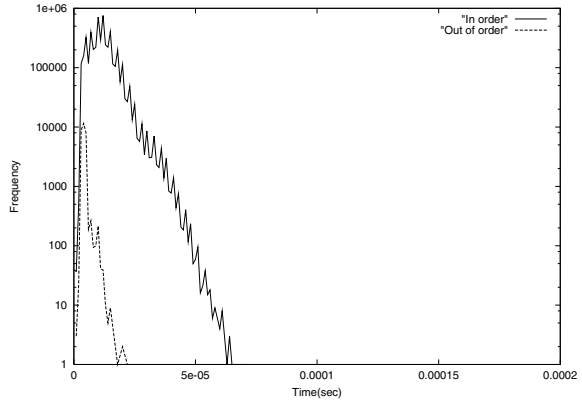
Consider the data sets which have the same fraction of reordered packets, but report different numbers of reordering events that would affect TCP. For example, F_{700}(LA→Pitt, 1500) contains 0.04% out of order packets according to metric O_1, but none of these events would affect a TCP flow (i.e. $O_2 = 0$). However, the same flow on the reverse path, F_{700}(Pitt→ LA, 1500), experiences $O_2 = 30$ reordering events that would affect TCP, even though O_1 measures the same fraction of out of order packets. Likewise, a comparison between F_{300}(LA→ Pitt, 500) and F_{300}(Pitt→ LA, 500) shows similar behavior: the two flows have approximately the same amount of reordering measured by metric O_1 (0.6%), yet yield vastly different numbers of reordering events when measured using the TCP-like metric ($O_2 = 0$ vs. $O_2 = 41$).

To further demonstrate the difference between the two metrics, we plot histograms of the frequency of packets delivered out-of-order by N places (where N reflects the difference in sequence number of the expected packet and the packet which did arrive at the destination). We chose two cases, illustrated in Figure 3: F_{600}(DC→LA, 1500) which contains significant packet reordering of adjacent packets only, with no reordering events that would affect TCP flows, and F_{300}(LA→ Pitt, 500) where many packets were delivered dozens of places out-of-order, in a way which would significantly impact the performance of a TCP flow.

It is clear that there are different reordering processes in operation for these two flows, and that these differences are not captured by the simple metric, O_1. This demonstrates the importance of choosing the correct metric to quantify packet reordering, particularly in terms of relevance to both the application and transport protocols used. It also makes it clear that, if one wishes to compare results of studies taken on different networks, or with different transport protocols, that a standard metric is needed. The IETF has ongoing work in this area [13] and our work serves to highlight the importance of this effort.

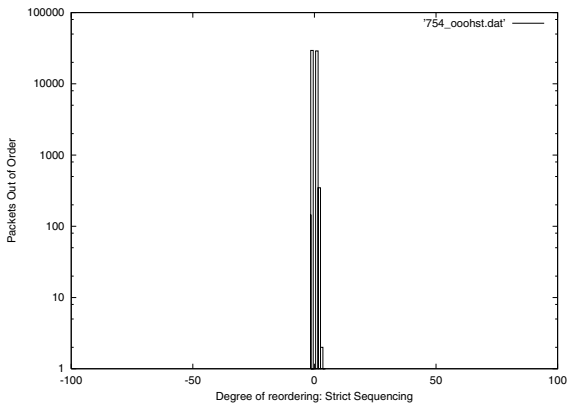


F_{800}(Pitt→ DC, 1500)

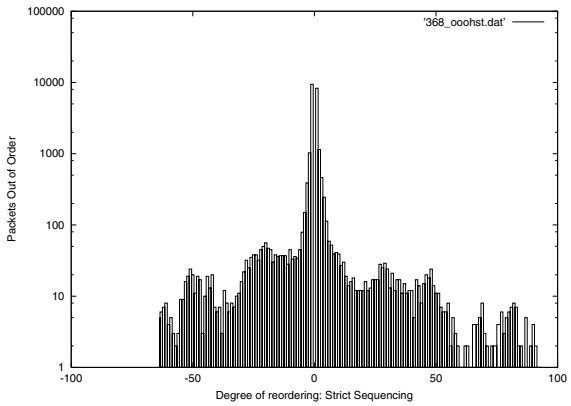


F_{300}(LA→ Pitt, 500)

Fig. 2. Packet inter-arrival times: In-order versus out-of-order



F_{300}(UDP, LA→ Pitt, 500): $O_1 = 0.59, O_2 = 0$



F_{600}(UDP, DC→LA, 1500): $O_1 = 0.71, O_2 = 3299$

Fig. 3. Degree of packet reordering

D. Discussion of Results

Our measurements have confirmed our initial hypothesis: the relative frequency of packet reordering increases as the inter-packet arrival time in the network core is reduced, so flows with high packet rates, or flows with closely spaced packets, will be more affected by reordering than low-rate flows.

As was shown in Table I and discussed in section IV-A, the raw rate of packet packet reordering in the networks we studied follows a pattern much like that seen in Figure 4. Clearly the particulars depend on network characteristics such as the type of infrastructure, routers and cross traffic, however as we discuss in the following, we do not believe it to be an uncommon pattern.

V. Implications for Transport Protocols and Networks

It seems likely that future high-performance networks will be implemented in a manner that allows packet reordering to occur and likely increase. The technical, economic and performance benefits of introducing parallelism into the network, be it through multipath IP routing, load balancing across layer 2 (and lower layer) paths, or striping packets across switch/router fabrics are too great to ignore. This will be further encouraged by the next generation of network performance evolution which is expected to be characterized by the introduction of parallel 10 Gbps links as opposed to an immediate jump to 40 Gbps technology. Parallelism is expected to be introduced at multiple levels of the network (layer 3, layer 2, optical), and points to a potential increase in packet reordering and an environment where the observed packet reordering is greatly effected by individual network architectures and transient traffic profiles.

These advances in network architectures and infrastructure are enabling high-performance applications from the scientific community such as high-energy physics, astronomy, geology

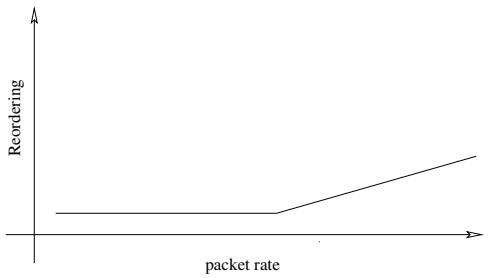


Fig. 4. Packet reordering versus inter-arrival times

and meteorology. In turn, these applications are initiating the need for Layer 4 transport protocols with features such as real-time reliable data delivery, multi-homing and multi-streaming, plug-in congestion control, etc.

This has resulted in the design of new transport protocols such as the Stream Control Transmission Protocol (SCTP) [17], the current work on the Datagram Congestion Control Protocol (DCCP) [12], and a number of modifications to TCP for improved performance. Suggested modifications to TCP range from making TCP more tolerant to packet reordering at the expense of reduced congestion response [3], modifying the TCP congestion response function for high speed connections that require large congestion windows [6] and variants of TCP such as FAST [4].

Until now packet loss has been treated as the main detriment to transport, and packet reordering as rare anomaly. Our data seem to indicate that reordering is on par, if not a larger issue, than packet loss on some classes of networks. Therefore protocols that treat a reordered packet equivalent to a lost packet will suffer performance consequences in such networks.

The prime example of this is TCP. Reordering affects the performance of a TCP flow because it treats certain patterns of reordered packets as a congestion signal. This problem permeates through the entire class of TCP-friendly protocols (DCCP and SCTP) and algorithms (TFRC) because, to be fair to TCP, they must emulate this behavior. Therefore, even moderate percentages of reordering make it difficult to sustain high data rates on modern networks. Indeed, on the network we tested, the limiting factor in TCP performance would have been spurious congestion signals caused by packet reordering, rather than actual packet loss. Therefore, for a TCP flow to sustain a high rate it must be able to tolerate the degree of steady state reordering we observed in our experiments.

We believe that there are many reasons to expect that reordering will remain prevalent and likely increase in future networks. Our results indicate that, in an environment such as this, transport protocol designers will need to explicitly consider the limits of performance caused by packet reordering. Reordering may have a direct impact on transport, as in the case of TCP, or indirectly, as is the case for FAST TCP, where packet reordering will vary round trip time computations and the congestion response.

We conclude that new protocol designs must pay heed to packet reordering, perhaps on equal footing to those caused by packet loss, in order to achieve high performance in future heterogenous networks.

VI. Summary and Conclusions

We conducted measurements of UDP/IP flows to study the occurrence of packet reordering. Our results demonstrate the high prevalence of packet reordering relative to packet loss and a strong correlation between packet rate and reordering on the network we studied. While we can speculate on the reasons for this, we note that in general reordering can happen due to a multitude of reasons. Our expectation is that the degree and probability for packet reordering will increase as modern networks continue to incorporate parallelism into network elements and architectures. Additionally, the advent of high bandwidth applications (such as HDTV and large e-science applications) will increase the instance of reordered packets on a per flow basis. New application and protocol design should treat packet reordering on equal footing as packet loss and be robust and resilient to both in order to achieve high performance.

VII. Acknowledgments

The authors would like to thank Nikhil Mittal for setting up the SCNM hosts and conducting many of the experiments. Our thanks also goes to Craig Leres, Brian Tierney, and Jason Lee of Lawrence Berkeley Laboratory for providing specification and configuration information for the monitoring hosts. This paper is based upon work supported by the DARPA Information Processing Technology Office and the National Transport Optical Network Consortium (NTONC). The opinions, findings, conclusions and recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of DARPA or NTONC.

References

[1] D. Agarwal and B. Tierney. Self-configuring network monitor project. http://www-itg.lbl.gov/Net-Mon/Self-Config.html.

[2] J. C. R. Bennett, C. Partridge, and N. Shectman. Packet reordering is not pathological network behaviour. *IEEE/ACM Trans. Networking*, 7(6):789–798, December 1999.

[3] E. Blanton and M. Allman. On making TCP more robust to packet reordering. *ACM Computer Communication Review*, January 2002.

[4] Jin C., Wei X. D., and Low H. S. FAST TCP: Motivation, architecture, algorithms, performance. In *Proc. IEEE Infocomm*, March 2004.

[5] DARPA. The NGI SuperNet testbed. http://www.ngi-supernet.org/.

[6] S. Floyd. High speed TCP for large congestion windows. IETF, August 2003. Work in Progress.

[7] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. of ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.

[8] L. Gharai and C. S. Perkins. Implementing congestion control in the real world. In *Proc. IEEE Intl. Conference on Multimedia and Expo*, Lausanne, Switzerland, August 2002.

[9] L. Gharai, C. S. Perkins, G. Goncher, and A. Mankin. RTP Payload Format for SMPTE 292M Video. IETF, March 2003. RFC 3497.

[10] M. Handley, J. Padhye, S. Floyd, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. IETF, January 2003. RFC 3448.

[11] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Measurement and classification of out-of-sequence packets in a Tier-1 IP backbone. In *Proc. ACM SIGCOMM Internet Measurement workshop*, Marseille, France, November 2002.

[12] E. Kohler, M. Handley, S. Floyd, and J. Padhye. Datagram congestion control protocol (DCCP). IETF, July 2004. Work in Progress.

[13] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, and J. Perser. Packet reordering metric for IPPM. IETF, Sept. 2003. Work in Progress.

[14] V. Paxson. End-to-end internet packet dynamics. *IEEE/ACM Trans. Networking*, 7(3):277–292, June 1999.

[15] C. S. Perkins, L. Gharai, T. Lehman, and A. Mankin. Experiments with delivery of HDTV over IP networks. In *Proc. 12th Intl. Packet Video Workshop*, Pittsburgh, April 2002.

[16] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. IETF, July 2003. RFC3550.

[17] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream control transmission protocol. IETF, October 2000. RFC2960.

[18] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. iperf. Software available online, October 2002. http://dast.nlanr.net/Projects/Iperf/.