Internet Traffic Distribution over *Multilink*

Where High Bandwidth Scalable Switch Port Aggregates Multiple Physical Links

Ju-Yeon Jo, Yoohwan Kim, and Frank L. Merat
Electrical Engineering & Computer Science Department
Case Western Reserve University
Cleveland, OH 44106

H. Jonathan Chao
Electrical & Computer Engineering Department
Polytechnic University
Brooklyn, NY 11201

Abstract – A logical link composed of multiple physical links is in extensive use in today's Internet and its use is growing due to good scalability, reliability and cost-effectiveness. When IP packets are distributed over such physical links, load unbalancing and packet reordering may occur. Since packet reordering degrades TCP performance, a good traffic distribution method must reduce the amount of reordering as well as packet loss. Hash-based load balancing is simple and ensures no packet reordering, but load unbalancing may occur. We have studied hash-based load balancing methods and analyzed their performances through simulation. We have created a simulation model with OPNET Modeler to generate massive TCP traffic, and then studied the load balancing performance and TCP throughput. The simulation results show that our proposed method, dynamic hashing with flow volume, has the best load balancing performance and minimum impact on TCP throughput.

Keywords—Multilink; Internet Traffic Distribution; Packet Reordering; Hashing; TCP Performance

I. INTRODUCTION

By bonding multiple physical interfaces together, a logical link with the aggregate bandwidth is formed, which performs and is administered as a single logical link as shown in Figure 1. Since a logical link requires only one entry in the routing table, it gives easier routing with exponential decrease in complexity than multiple entries. A logical link also simplifies management/operation and allows the Internet service providers to build scalable and reliable networks and reduce equipment and link costs. A logical link benefits the switch design in that switch port speed is not constrained by the limitations of the fiber infrastructure. Switch port speed is becoming higher surpassing the growth in link speed. A logical link technology allows a switch port to be connected to multiple physical links transparently to the switch. With such benefits, logical links are widely used in Internet backbone links[3].

However it is difficult not only to manage such parallel links at IP layer but also to efficiently utilize the links. In order to accommodate this demand, technologies that efficiently aggregate many individual links into a single data stream are getting investigated and deployed. Avici's *Composite Link*[2] and Pluris's *IP-Bond*[1] are the examples at core router. Similar concepts of bonding multiple physical links into one logical link are found elsewhere, such as *parallel communications link* of BBN[16], *link aggregation* of IEEE 802.3ad[8], *multilink* of PPP-MP[11], and *hunt group* of DEC Gigaswitch[3], etc. In this article, we adopt the term *multilink* to represent the concept of logical link composed of multiple parallel links.

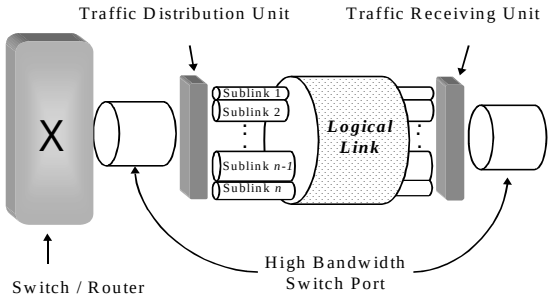

Figure 1: Multilink

The concept of *multilink* should not be confused with multipath routing, which is performed by the routing protocol at network layer. On the other hand, *multilink* is for point-to-point data delivery at data link layer. The traffic overload at network layer is handled by conventional methods such as WRED, which makes sure that the total traffic amount does not exceed the logical link capacity. *Multilink* handles the data stream that already went through such a congestion control mechanism. Although the incoming traffic amount to *multilink* does not exceed the combined capacity, it is still possible to have an unbalancing among the sublinks while distributing the data stream over multiple sublinks. The congestion control at network layer may not help such local congestion at *multilink* since it is only concerned on the satisfying the logical link capacity as a whole. Even if the network layer congestion control can help relieving the local congestion, the time granularity, typically seconds, is much greater than the required response time at *multilink*, typically milliseconds. Therefore *multilink* must be able to handle its own local congestion without asking for any correction from the sending party.

Our research is focused on the efficient traffic distribution over *multilink* while minimizing the packet loss caused by sublink buffer overflow and the out-of-order packet delivery that affect the TCP performance. It has been known that such packet reordering results in many performance penalties in TCP and should be avoided as much as possible[3][17]. For

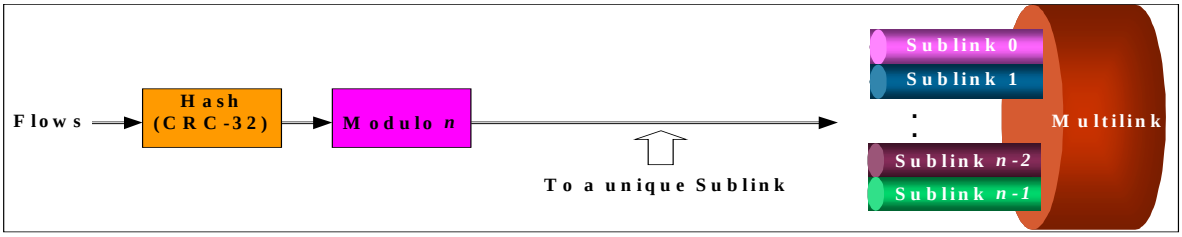

Figure 2: SH (1-Stage)

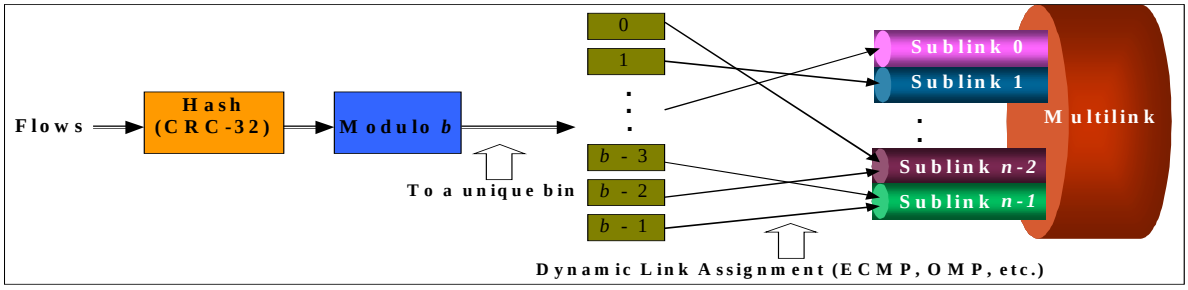

Figure 3: DH (2-Stage)

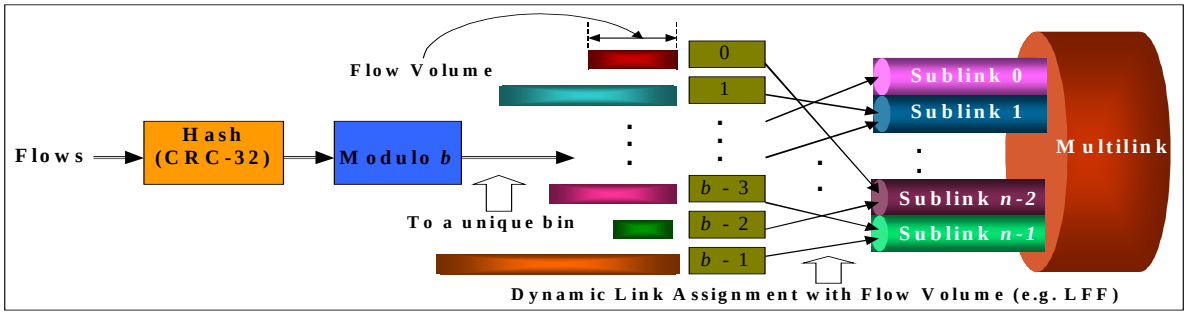

Figure 4: DHFV (2-Stage)

ensuring in-order packet delivery, hashing is a preferred technique because of its simplicity. Hashing the invariant fields in the packet header generates a unique number for each flow, so the packets belonging to a flow can be sent over a unique sublink, thus reordering is avoided. However, static hashing (SH) alone does not guarantee a good load balancing. Dynamic hashing (DH) can fix the unbalancing by reassigning some flows, but it introduces a large amount of packet reordering. Our proposed method, dynamic hashing with flow volume (DHFV) [10] detects the largest flow and reassigns only one flow and shows reduced packet reordering as well as excellent load balancing performance. In this paper, we verify the load balancing performance and examine the end-to-end TCP performance, which is the ultimate goal of traffic distribution methods.

II. LOAD BALANCING BY HASHING

A. *Obtaining a hash value from a packet*

A flow is a stream of packets sent from an application at a source machine to another application at another machine while the application is active. For the packets belonging to the same flow, there are invariant fields in the header and hashing them results in a unique flow ID.

For all load balancing methods, a hash value is generated as following.

Hash value = *H* (Header invariant fields) modulo *k*,
where *H = hash function*,
k = number of sublinks for static hashing, or
k = number of bins for dynamic hashing

For *H*, we use CRC-32 hashing in all our simulation, which is known to have a very good performance, [4][5][9] with 96-bit input as shown in Figure 5.

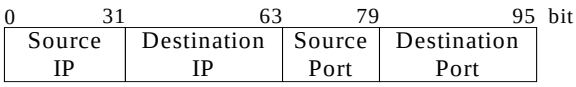

Figure 5: Hash Input Fields

1)Static hashing (SH)

With SH, a particular flow is assigned to a unique sublink and the assignment is not changed during operation. SH uses the hash value directly to assign a flow to a sublink. Figure 2 illustrates SH.

2)Dynamic hashing (DH)

DH has an additional step from SH. The hash value is first used to determine an intermediate bin number, and then used next to determines the sublink number as shown in Figure 3. DH allows load balancing by changing the number of assigned bins to each sublink. Periodically the queue length of each sublink is monitored to determine if any sublink is overloaded.

In case of overload, a number of bins are reassigned to a lighter traffic loaded sublink.

3)DHFV

DHFV is an enhanced version of DH, where the flow volume for each bin is measured and used to determine the *bin-to-sublink* assignment more intelligently as shown in Figure 4. Flow volume is the amount of traffic assigned to a bin per unit period. Since each bin is an approximation of a flow, it is considered flow volume.

Among the bin selection strategies, it is know that Largest Flow First (LFF) method works best, where the bin with the largest flow volume from the overloaded link is moved to the underutilized link. It has been shown that there is a high temporal locality in flow volume for large flows [10]. That is, the flow that had the largest flow volume in the past usually produces fairly large flow volume in the future, so moving such a flow reduces the bandwidth in the congested sublink very effectively. As in DH, the queue length is monitored periodically for detecting overloaded link. If there is any overloaded link, LFF algorithm is activated.

III. Simulation Model

Our earlier study of packet trace data shows that DHFV-LFF performs best in load balancing [10]. In this paper, we create a simulated network environment with OPNET Modeler simulation software and observe whether the methods perform similarly in an interactive traffic environment as in off-line case. Further we would like to know how the packet loss and reordering affect the TCP traffic characteristics and which method achieves the best performance in TCP throughput. Since the TCP traffic accounts for the majority of the Internet traffic and the TCP performance is the ultimate user-perceived performance, TCP performance is more meaningful than local load balancing performance. We now build a simulation model of *multilink* and apply a similar traffic as in real Internet.

A. Network environment

We create two subnets connected by a logical DS3 link. The logical link is divided into 8 physical sublinks with equal bandwidth. The propagation delays of the sublinks are identical, but still the packet reordering will occur as a result of queue length difference for each sublink. There are 1,200 hosts on client side and 4 servers on server side. The majority of the traffic moves from the server side to the client side, and the traffic for that direction is divided into sublinks. More detailed OPNET simulation environment are described in authors' another paper[11].

B. Traffic generation

In Internet traffic, it is known that small number of large flows occupy big portion of the traffic. [6][13][18] In our trace data, obtained from NLANR project [15], only 0.1% of flows account for 53% of the traffic amount and less than 1% of flows occupied 80%. This bias causes the SH to perform poorly. To simulate it, we generate small number of large ftp traffic and large number of small traffic. The types of traffics are,

- FTP: exponential distribution with mean of 600 K bytes
- HTTP: exponential distribution with mean of 10 K bytes
- Email and DB: exponential distribution with 3 K and 2 K bytes, respectively

The above types of traffic are generated by the client machines, making about 50,000 flows during 20 seconds. This simulated traffic generates a similar traffic to the packet trace data with average link utilization rate of 40.45% and peak rate of 79.9% and no packet loss. The data on the undivided logical link, denoted as a single link, will be used for a comparison with *multilink*. Our goal is achieving the performance on *multilink* as close as possible to single link.

C. Details of DH and DHFV implementation

In DH and DHFV, the sublink queue lengths are checked in every 25 milliseconds. If there is any sublink buffer(s) that has exceeded 50% of the buffer capacity, some of the bins belonging to that sublink are reassigned to another sublink. Specifically, 10% of the bins are reassigned to another sublink in DH, and only one bin with the largest flow volume is reassigned in DHFV. If the load-balancing interval (25ms) is reduced, the load balancing effect becomes better, but the amount of reassignments may increase.

In DHFV, the flow volume for each bin is collected in every 10 ms by adding the sizes of assigned packets. To eliminate the fluctuation in measurement, we add multiple periods of measurements. Before picking up the largest flow volume, 10 of the most recent measurements are added and then compared, so the flow volume reflects the most recent 100 ms of traffic.

One important procedure in both DH and DHFV is the queue growth trend monitoring to reduce unnecessary reassignments. Even if the buffer occupancy is greater than 50%, no more reassignment is done if the queue length is decreasing. This is because the decreasing queue length indicates that the previous reassignments became already effective.

D. Parameter settings

Other parameters are configured as following.

- Number of sublinks: 8
- Sublink bandwidth: 5,592,000bps (= DS3 bandwidth/8)
- Propagation delay for all sublinks: 1 ms
- TCP version: Reno with SACK option
- Buffer size: 559,200 (100 ms capacity of each sublink bandwidth)
- Number of bins: 1024
- Simulation duration: 20 seconds

IV. Simulation Results

A. Traffic distribution

The Figures 6 - 8 show the traffic amount in bits that are sent through each sublinks. In SH, buffer overflow (flat lines on 5,592,000) in a few sublinks are observed.

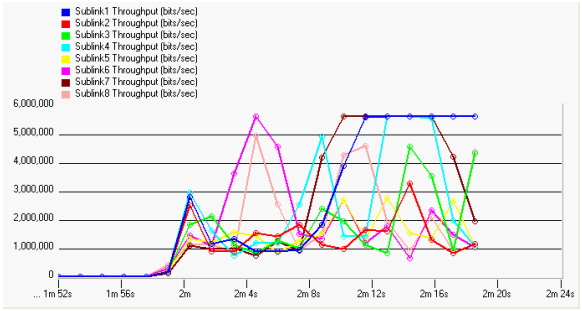

Figure 6: SH

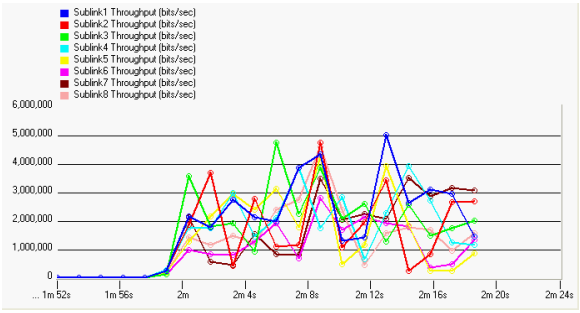

Figure 7: DH

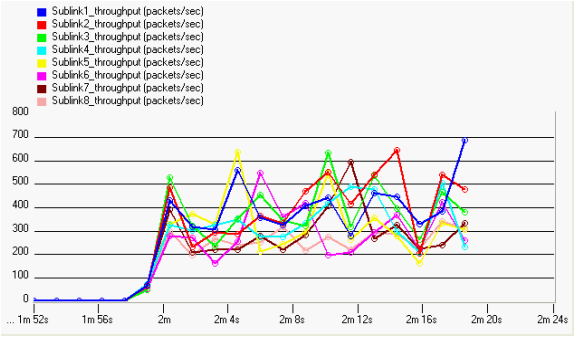

Figure 8: DHFV

B. Lost traffic

The Figure 9 shows the amount of lost packets during simulation time of 20 seconds, which is caused by buffer overflow. The loss is significant for SH (352K bytes) while DHFV (11K bytes) and DH (39K bytes) have less loss.

C. Number of sublink reassignments

The sublink reassignment causes packet reordering. So the number of reassignment is another factor to affect the TCP performance. Obviously there is no reassignment with SH, and DHFV introduces more than 60 times less reassignments than DH as shown in Figure 10.

D. TCP end-to-end performance

In single link environment, the amount of transmitted data is same as the amount of received data because there is no loss and no retransmission. But in SH, DH, and DHFV, there are losses and reordering, so senders' congestion windows are decreased and retransmissions occur. It increases the bandwidth usage, amount of total transmission, and the time for completing transmission.

Figure 11 shows the link utilization as a whole. It shows that no load balancing method can achieve the same performance as the single link.

All methods finish the transmission eventually if enough time is given. So we take a look at the sent and received traffic amount for a fixed time. Figure 12 shows the amount of total sent traffic and received traffic at TCP layer during 20 seconds. Due to retransmission, the amount of received traffic is always smaller than the amount of sent traffic. The TCP throughput at single link shows the maximum achievable throughput. (The difference reflects the packets in the sublink queues at the moment of simulation end time.)

DHFV shows a very close performance to single link. It performs much better than DH because DHFV does not introduce as much packet loss or sublink reassignments as DH. In DH, the amount of transmitted data is reduced due to reduced TCP congestion window caused by packet reordering. SH caused a lot of retransmission due to packet loss, so it shows inefficient link usage.

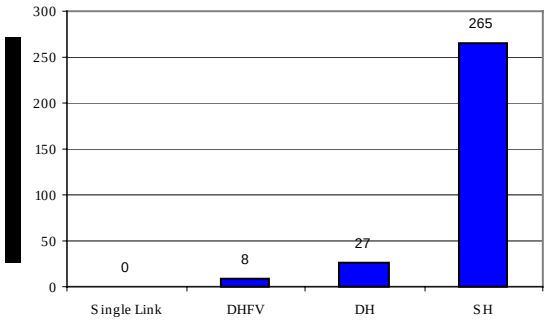

Figure 9: Lost packets

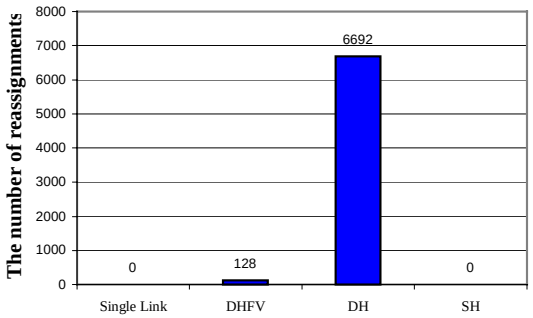

Figure 10: Sublink reassignments

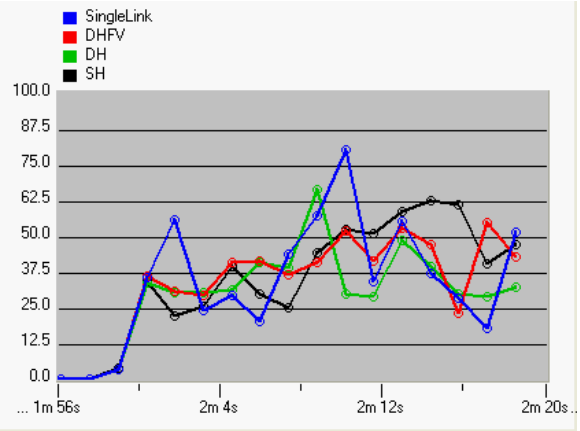

Figure 11: Link Utilization

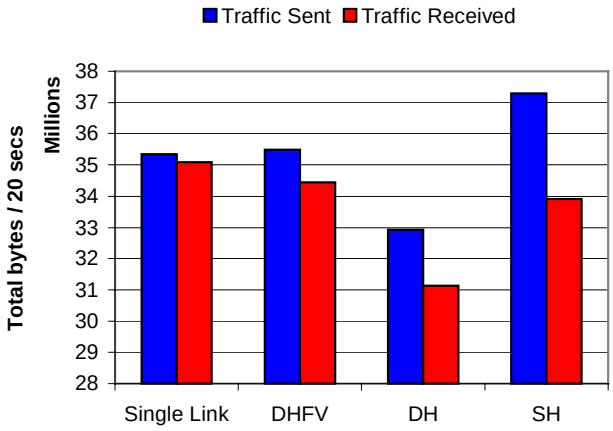

Figure 12: Total TCP Traffic during 20 secs.

E. Discussions

Further simulation shows that the unbalancing becomes greater as the number of sublinks increase. While the link utilization is low, the performance difference is not visible, but as the link utilization is increased, the loss in SH and reassignment in DH become greater and DHFV performs better.

The performance advantage of DHFV comes from the flow bandwidth distribution in Internet traffic, where small number of big flows account for the majority of the traffic. The sublink buffer overflow is usually caused by the big flows, so without reassigning the big flows, load balancing cannot be effective. While DH tries to select such flow by randomly selecting multiple flows, DHFV precisely selects such large flows. If all the flows in Internet traffic have the same flow bandwidth, DH would perform best while DHFV would fail.

In current TCP implementation, the performance drop by packet reordering is significant. To reduce such penalty, an extended version of SACK[14] option, DSACK[7] is standardized. Still some performance penalty is unavoidable once packet reordering occurs. So it is still desirable to reduce the amount of packet reordering as much as possible.

V. CONCLUSIONS

Multilink is being employed more extensively and hashing is considered a cost-effective way for traffic distribution to avoid packet reordering. To achieve load balancing, DH may be used, but due to excessive reordering, the TCP performance drops. An improved algorithm, DHFV reassigns flows selectively to achieve load balancing by reassigning minimum amount of flows. With DHFV, the aggregated TCP performance is very close to the performance of single link.

With OPNET Modeler, we have generated application traffic to emulate the Internet packet trace data, and implemented the load balancing algorithms. The simulation results show that DHFV achieves very good load balancing with minimum impact on TCP throughput.

References

[1] Vadim Antonov and David Bernstein, “The Pluris Massively Parallel Router (MPR),” *Gigabit Networking Workshop*, 1998.

[2] Avici White Paper, “The Use of Composite Links to Facilitate the Creation of a New Optical Adaptation Layer,” http://www.avici.com.

[3] Jon C. R. Bennett, Craig Partridge, and Nicholas Shectman, "Packet Reordering is Not Pathological Network Behavior," *IEEE/ACM Transactions on Networking*, vol. 7, No. 6, pp. 789-798, December 1999.

[4] Zhiruo Cao, Zheng Wnag, and Ellen Zegura, "Performance of Hashing-Based Schemes for Internet Load Balancing," In *Proceedings of IEEE Infocom*, pp. 332-341, 2000.

[5] Girish Chandranmenon and George Varghese, “Trading Packet Headers for Packet Processing,” *IEEE/ACM Transactions on Networking*, vol. 4, No. 2, pp. 141-152, 1996.

[6] Anja Feldmann, Jennifer Rexford, and Ramon Caceres, “Efficient policies for carrying Web traffic over flow-switched networks,” *IEEE/ACM Transactions*, Vol. 6, No 6, pp. 673-685, Dec. 1998.

[7] Sally Floyd, Jamshid Mahdavi, Matt Mathis, and Matt podolsky, “An Extension to the Selective Acknowledgement (SACK) Option for TCP,” July 2000. RFC 2883.

[8] IEEE Standard 802.3ad-2000, Link Aggregation (CSMA/CD: ETHERNET)

[9] Raj Jain, "A Comparison of Hashing Schemes for Address Lookup in Computer Networks," *IEEE Transactions on Communication*, vol. 40, pp. 1570-1573, Oct. 1992.

[10] Ju-Yeon Jo, Yoohwan Kim, H. Jonathan Chao, and Frank Merat, “Internet Traffic Load Balancing using Dynamic Hashing with Flow Volume,” *Proceedings of SPIE ITCom*, July 2002.

[11] Ju-Yeon Jo, Yoohwan Kim, and H. Jonathan Chao, “TCP Performance Comparison under Various Load Balancing Methods using OPNET” *OPNETWORK 2002*, Aug. 2002.

[12] K. Kslower, B. Lloyd, G. McGregor, D. Carr, and T. Coradetti, “The PPP Multilink Protocol (MP)”, Aug. 1996 RFC 1990.

[13] Ratul Mahajan, Sally Floyd, and David Wetherall, “Controlling High-Bandwidth Flows at the Congested Router,” In Proceedings of International Conf. On Network Protocols (ICNP), November 2001.

[14] Matt Mathis, Jamshid Mahdavi, Sally Floyd, and A. Romanow, “TCP Selective Acknowledgement Option,” Oct. 1996 RFC 2018.

[15] NATIONAL LABORATORY FOR APPLIED NETWORK RESEARCH (NLANR). Network traffic packet header traces. http://moat.nlanr.net/Traces/Traces/daily/20011204/IND-1007501453-1.tsh.

[16] Craig Partridge and Walter Milliken, “Method and Apparatus for Multiplexing Bytes Over Parallel Communications Links Using Data Slices”, *US Patent 6,160,819*, Dec. 12, 2000.

[17] Vern Paxson, “End-to-End Internet Packet Dynamics”, *IEEE/ACM Transactions on Networking*, vol. 7, No. 3, pp. 277-292, June 1999.

[18] Carey Williamson, “Internet Traffic Measurement,” *IEEE Internet Computing*, Nov./Dec. 2001, pp. 70-74.