WPI-CS-TR-04-13 May 2004

Exploiting Flow Relationships to Improve Performance of Networked Applications

by

Hao Shang
Craig E. Wills

Computer Science
Technical Report
Series

WORCESTER POLYTECHNIC INSTITUTE

Computer Science Department
100 Institute Road, Worcester, Massachusetts 01609-2280

Abstract

The use of networked applications on the Internet is increasing, both in the number of applications and the amount of network flow traffic they generate. These applications include the Web, streaming media, games, peer-to-peer and grid computing where each application generates its own network flow dynamics between nodes in the Internet. A single application often causes other network flows, such as DNS and authentication, apart from those that it generates directly. With the ever-increasing number and variety of network applications available, an interesting, but unexplored direction for research is understanding the relationships among network flows between hosts and sites on the Internet and how these relationships can be exploited for improved application performance.

In this work, we present results on the degree to which relationships between network flows exist between host and site pairs. We go on to study relationships for flows of specific network applications. We use these relationships as a basis to propose a new approach for packet transmission using an "active network layer" where packet transmissions from respective transport layer protocols are passed down to this active network layer with a deadline for transmission. The inclusion of a deadline allows the network layer to perform real-time scheduling from a pool of packets and to encapsulate multiple less-than-full packets into the same transmitted frame thus exploiting concurrent flow relationships while not introducing additional frames that need to be routed. The availability of this network layer also allows the possibility of speculative packet transmission when these packets can be combined with other network traffic for improved reliability of applications without introduction of additional transmitted frames.

1 Introduction

With an increasing number and variety of networked applications on the Internet there is opportunity to examine the relationships between the network flows of these applications to improve network and application performance. In this work we study the relationships among network flows between hosts and between clusters of hosts.

We define a *flow* as the collection of all packets over a period of time using the same transport protocol (e.g. TCP, UDP) between source and destination hosts where all packets share the same source and destination port. For a TCP flow, it is normally bounded by SYN and FIN/RST packets. For flows of other types including UDP, the end of a flow is decided when a period of idle time (we use five minutes in our experiment) is observed. To facilitate our study, we define a flow to include traffic in both directions.

We define a *relationship* to exist between two flows if the flows exhibit temporal proximity within the same scope. The scope may either be between two hosts or between two clusters of hosts where a *cluster* is the set of hosts at a site sharing the same end router. We define a relationship to be *concurrent* if the beginning of a flow coincides with an active flow in the same scope. We define a relationship to be *sequential* if the beginning of a flow follows a recently concluded flow in the same scope.

Relationships occur between the flows of networked applications for a number of reasons:

1. Application behavior. One example of a host-to-host concurrent relationship is when a streaming application creates separate network connections for control and data flow. Another example of a cluster-to-cluster sequential relationship is when the application network flow is preceded by a DNS (Domain Name System) lookup causing a network flow between a local DNS server and a remote authoritative DNS server.

2. Content relationships. An example is when multiple servers at a Web site serve content for a page leading to concurrent cluster-to-cluster network flows when a Web browser downloads the page content.

3. User behavior. Cluster-to-cluster relationships occur when multiple users at a site are interacting with the same remote server such as playing a network game using the same game server.

With these definitions of relationships and enumerated cases where they exist, we study the extent that flow relationships are observed in current network traffic and measure the flow relationships of specific networked applications. In light of these relationships and the fact that 70-90% [25, 8] packets are smaller in size than the MTU (maximum transmission unit) of the network, we propose a novel approach to change how network transmission is currently done. We explore the introduction of an "active network layer" where packet transmissions from respective transport layer protocols are passed down to this active network layer with a deadline for transmission. The inclusion of a deadline allows the network layer to perform real-time scheduling from a pool of packets and to encapsulate multiple less-than-full packets into the same transmitted frame thus exploiting concurrent flow relationships and reducing the number of frames to be routed.

Such an active network layer could be used for transmissions between two hosts or could be used to pool all packets for transmission from one cluster to another. The availability of this network layer also allows the the possibility of speculative packet transmission when these packets can be combined with other network traffic to the same destination host or cluster. The possibility of piggybacking additional packets on existing traffic affords applications the capability of sending duplicate copies of important packets for better reliability.

As part of presenting this approach, we describe specific examples of how this approach can reduce the number of round trips between sender and receiver, remove packet exchanges from the critical path of application response, and improve the reliability of network applications. We go on to describe the current status of this work. We conclude with a discussion of related work that has used techniques to exploit flow relationships and finish with a summary of our work to date.

2 Existence of Flow Relationships

Our initial work examined the extent to which relationships exist among network flows between the same hosts and the same clusters. For this work we used the set of hosts on the WPI campus as a cluster and obtained logs of network flow data to and from the WPI campus network for a full day on July 7, November 30 and December 17, 2003. The tool argus [1] was used to trace IP packets and combine these packets into flows based on common host, port and transport protocol. For the analysis we focused on TCP and UDP flows with roughly 80% of these flows for TCP traffic. The July log was collected during summer vacation and contained only 3.9 million flows while the November and December logs have 12.3m and 15.5m flows, respectively.

We examined the percentage of flows for each of the three days in which a flow follows a previous flow between the same host pairs using different time thresholds. The results are shown in Table 1. The threshold of zero seconds in the table indicates a concurrent flow where an existing flow between the same hosts exists when a new flow begins. The larger threshold values include flows that begin within the given time interval after a previous flow between the same hosts. We see a significant relationship between flows even with thresholds as small as 10 seconds.

Table 1: Percentage of Host-to-Host Network Flows within a Specified Time Threshold of a Previous Flow

Time	Threshold			
Period	0 Sec	10 Sec	30 Sec	180 Sec
Jul ’03	28.2%	47.2%	51.2%	61.0%
Nov ’03	38.5%	53.7%	58.6%	68.3%
Dec ’03	27.4%	41.6%	46.4%	56.9%

We also studied the same data by grouping the set of non-WPI hosts into clusters (all WPI hosts form a single cluster). Ideally, we would have used a clustering tool based on BGP routing information as done in [14], but such a tool is not available in the public domain. Consequently we used an approximate classification defined by traditional Class B and Class C addresses. While we know such an approach is not accurate in all cases, we believe it is good enough when examining traffic with a common end-cluster (WPI) over narrow windows of time. We used this approach to gain an understanding on the number of potential relationships that exist on a cluster-to-cluster basis. These results are shown in Table 2.

Table 2: Percentage of Cluster-to-Cluster Network Flows within a Specified Time Threshold of a Previous Flow

Time Period	Threshold			
	0 Sec	10 Sec	30 Sec	180 Sec
Jul '03	47.5%	68.4%	73.1%	82.5%
Nov '03	60.9%	76.3%	80.4%	87.1%
Dec '03	52.8%	78.2%	82.0%	89.1%

These results show that more than half of flows exist in parallel with other flows from the same cluster and three-fourths of flows exist within 10 seconds of a previous flow from the same cluster. Since the last two logs are collected when school was in session and include more network flows, they show more relationships than those in the first log in the cluster-to-cluster scope.

3 Relationships for Specific Applications

The results in Tables 1 and 2 show the existence of a significant number of relationships among network flows between hosts and clusters. We further broke down the network flows according to their related applications and studied the relationship between each type of flows. We found the relationships between different types of flows to be relatively stable and to exhibit a similar pattern for all three logs. Table 3 shows the results for a small sample of applications from the December 2003 log.

The first column in Table 3 is the application type based on transport protocol ("t" for TCP and "u" for UDP) and port number. Columns 2 and 3 show related flows that exist between two hosts for thresholds of 0 seconds (concurrent flows) and 30 seconds (a previous flow existed within the last 30 seconds). Similarly, columns 4 and 5 show related flows that exist between hosts in two clusters. We again show results for a threshold of 0 and 30 seconds.

The results shown in each cell of the table are the percentages of flows for the flow type in the first column that are related to other types of flows (including its own type). To conserve space we only list specific flow types when the relationship occurs for more than 10% of flows. In addition, we show cumulative percentages for all TCP and UDP flows.

The results in Table 3 show a number of relationships. A FTP (File Transfer Protocol) or SSH (Secure SHell) flow follows a previous flow of the same type in about 15% of cases within the same host pairs. The percentages become much larger within the scope of the same cluster pairs, where a FTP flow starts within 30 seconds of another FTP flow for over 90% of cases. Other types of flows like security key exchange (using UDP port 500), Web (using port 80 and 443), are also often observed before a FTP or SSH flow within the same cluster pairs.

The authentication protocol using TCP port 113 is used 86.9% of the time with the SMTP protocol (t25). The DNS protocol (u53) precedes all applications when we consider cluster-to-cluster sequential relationships. HTTP flows (t80) are used 43% of the time concurrent with a previous HTTP flow between the same hosts and over 60% of the time concurrent with hosts in the same cluster. The last column shows that over 90% of HTTP flows occur within 30 seconds of a previous HTTP flow between hosts in the same cluster.

Table 3: Relationship between Selected Flow Types

AppPort	h2h:0s	h2h:30s	c2c:0s	c2c:30s
t21 (ftp)	tcp:3.3% udp:0.0%	t21:13.8% tcp:14.5% udp:0.0%	t21:63.1% tcp:64.7% udp:14.3%	t113:26.8% t21:93.7% tcp:95.0% u500:11.2% udp:25.5%
t22 (ssh)	tcp:9.1% udp:0.1%	t22:15.9% tcp:17.0% udp:0.1%	t22:24.7% tcp:44.0% u500:12.5% udp:18.5%	t110:18.1% t22:33.5% t443:14.6% t80:12.1% tcp:57.5% u500:15.1% udp:23.6%
t25 (smtp)	tcp:5.2% udp:0.0%	t25:18.0% tcp:18.1% udp:0.0%	t25:11.7% tcp:15.4% udp:9.4%	t25:30.0% tcp:34.9% u53:15.8% udp:16.9%
t80 (http)	t80:43.0% tcp:43.1% udp:0.1%	t80:58.6% tcp:58.7% udp:0.1%	t80:61.1% tcp:62.2% u53:10.0% udp:11.4%	t80:91.9% tcp:92.4% u53:14.1% udp:15.7%
t113 (auth)	t25:86.9% tcp:98.4% udp:0.0%	t25:86.9% tcp:98.5% udp:0.0%	t25:87.7% tcp:99.2% udp:5.3%	t113:11.2% t25:87.8% tcp:99.7% u53:13.7% udp:15.5%
t554 (rtsp)	t554:10.5% tcp:13.8% udp:4.5%	t554:53.1% tcp:55.7% u6970- 7170:10.4% udp:11.8%	t554:47.3% t80:16.8% tcp:63.7% u6970- 7170:38.8% udp:49.5%	t554:72.6% t80:23.0% tcp:82.2% u53:16.4% u6970- 7170:42.6% udp:60.8%
t7070 (real-stream)	t554:40.0% t7070:36.0% t80:68.0% t8080:56.0% tcp:84.0% udp:0.0%	t554:76.0% t7070:72.0% t80:82.0% t8080:76.0% tcp:86.0% udp:0.0%	t554:40.0% t7070:36.0% t80:86.0% t8080:56.0% tcp:92.0% udp:2.0%	t554:76.0% t7070:74.0% t80:92.0% t8080:76.0% tcp:92.0% u53:10.0% udp:10.0%
u6970-7170 (real-stream)	t554:53.9% tcp:54.5% udp:6.7%	t554:54.0% tcp:54.6% u6970- 7170:32.4% udp:34.2%	t554:54.0% tcp:54.7% u6970- 7170:42.6% udp:45.2%	t554:54.1% tcp:54.9% u6970- 7170:69.6% udp:74.5%
u27015-27017 (halflife)	tcp:0.0% udp:2.8%	tcp:0.0% udp:5.1%	tcp:1.5% u27015- 27017:11.0% udp:12.3%	tcp:2.2% u27015- 27017:30.9% udp:32.2%
u41170 (blubster)	tcp:0.0% udp:1.2%	tcp:0.0% u41170:13.0% udp:13.0%	tcp:5.1% udp:2.6%	tcp:7.3% u41170:15.5% udp:16.8%

Real stream application normally uses multiple flows. The control flow using RTSP protocol (t554) is frequently seen in parallel with the data flows using either TCP (t7070) or UDP (u6970-7170). We also observe the TCP-based Real player streaming frequently occurs in temporal proximity to HTTP.

"halflife", a popular on-line game, is used often concurrently or sequentially by different hosts, but within the same cluster. Network flows generated by a peer-to-peer file sharing application "blubster" have a certain amount of sequential relationships to its own type.

4 Application Traffic Behavior

The application-specific results again show that significant relationships do exist between network flows, although the results do not reflect the specific traffic pattern of packets within a flow, which is not available in the log. Knowledge of specific traffic behavior is important as we look at exploiting the relationships between network flows. We studied the packet and flow behavior for a number of sample applications such as ssh, the Internet Explorer browser, RealAudio, RealVideo, Windows Media Player, QuickTime, network games, instant messaging and electronic mail applications. From the results of this study we make a number of observations about the behavior of applications:

- A single application often causes multiple flows to be created to the same host or hosts within the same cluster.
- Interactive applications such as ssh, games and instant messaging generally use small packets. Applications using TCP use small packets in setting up a connection and sending acknowledgments.
- Many applications that use TCP have the PUSH flag set if the packet has a less-than-full-MTU, even if the packet may not need to be sent immediately to avoid control by Nagle's algorithm, which attempts to aggregate small amounts of TCP data [16]. The setting of the PUSH flag causes the data to be immediately sent.
- The packet size for streaming applications depends on the encoding, but most packets we observed are not full. When TCP is used for streaming, the server always uses the PUSH flag.

While these observations are not novel, each of these is a factor as we examine exploiting the relationships between flows. In the following we describe an approach that takes these observations into account.

5 Proposed Approach

The observation that many transmitted packets are less than full (previous studies have found 70-90% of packets are less than 1500 bytes [25, 8]) and because the cost of routing packets on the Internet is dominated by the number of packets and not the size, we are motivated to explore an approach that makes use of this unused capacity in packet transfer. As technology improves and network MTUs grow larger this wasted capacity will become more pronounced.

The approach of aggregating packets for delivery is an obvious direction to explore, particularly given the significant number of related flows. The natural point to do such aggregation is at the network layer—whether on a single host or at an end router for a cluster of hosts. However, an inhibiting problem with current transmission design is that applications and the transport layer have no means of specifying deadline constraints on when data must be sent. UDP data is always sent immediately while by default TCP transmitted data is collected and sent when a full packet is available or delayed until all transmitted data has been ACKed. For a TCP application to avoid this delay, it must use the PUSH flag to send the data immediately, which as observed is often used by applications. Without a means for specifying appropriate deadline constraints there is no "pool" of packets that can be potentially aggregated for delivery.

As an alternate approach to the traditional transmission mechanism, we propose a new approach as shown in Figure 1 where in addition to the packet data, the transport layer passes a deadline to specify the latest that the network layer can transmit a packet. The network layer then becomes an active entity that schedules packet transmission based on deadlines.

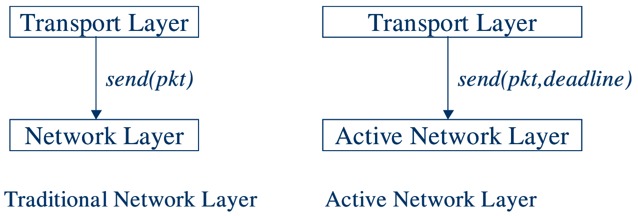

Figure 1: Addition of a Deadline Requirement When Sending a Packet

The deadline value can be set by either the application or transport protocols depending on their particular needs. For example, an application such as ssh may set an immediate deadline, but because its packets are small there is much potential for piggybacking additional packets in the transmitted frame. TCP SYN packets also would have an immediate deadline for delivery, but are small. A real-time stream application may be willing to tolerate a fixed lag in the delivery of its packets to allow possible aggregation with other traffic (possibly its own). A file transfer application may set all its packets with a much larger deadline indicating the traffic is not urgent.

The active network layer maintains a pool of packets and deadlines, using real-time scheduling to transmit the packets. In the case of less-than-full packets, the network layer can potentially aggregate other waiting packets. Delivery of such aggregated packets could be done through encapsulation of multiple IP packets into a single IP packet as is currently done for tunnelling [18]. This approach also preserves upper-level data boundaries as advocated by Clark and Tennenhouse [4].

The availability of such an approach affords a number of opportunities:

1. Acknowledgment mechanisms are an obvious use of this approach. When a packet arrives an acknowledgment can be immediately sent with a deadline corresponding to a desired ACK time-out. Such a time-out mechanism is built-in to the TCP protocol, but this approach would handle acknowledgments in a general way and take advantage of other traffic.

2. The introduction of deadline-based transmission gives applications more precise control over when data is sent. Under current protocols data can only be sent immediately or with TCP it is buffered without control by the application on exactly when it is sent.

3. Use of the approach below the transport layer allows independent, but related flows to be aggregated. Data from scenarios such as playing of an audio stream with one protocol concurrent with delivery of Web objects can be potentially aggregated.

4. The approach allows interactive and non-interactive traffic to naturally aggregated where packets for interactive traffic are often small, but have immediate deadlines while packets for non-interactive traffic are often larger, but have relaxed transmission deadlines.

6 Extension of the Approach

While potential aggregation of packet traffic is an attractive feature of this approach, it can be extended to allow additional features for upper layers. We also propose to include a "hard/soft" flag argument to the *send()* function in Figure 1 indicating whether the transmission must occur or should occur if possible. A "hard" value indicates a normal transmission with the given deadline. However a "soft" value indicates the packet will only be sent if it can be piggybacked. The introduction of a "soft send" allows for improving performance and reliability of applications and transport protocols without introducing additional transmissions.

One scenario of using a soft send is to protect critical packets by sending them in duplicate. For example, the loss of TCP SYN packet can have a more negative impact than the loss of a regular packet in the middle of the connection. In order to improve reliability, one approach is to soft send a duplicate SYN packet after the original SYN packet is sent. Another example of improving application reliability is for a streaming application to soft send duplicate copies of important packets of data. Duplicate DNS requests could be soft sent with a shorter timeout than normal where the DNS application can retract the soft send if a response for the original request is received.

The use of a soft send can also be combined with prediction. For example, a Web browser may predict the need of an additional TCP connection and use the soft send feature to set it up. The active network layer needs to notify the upper layer if a packet requested for a soft send is actually sent.

7 Current Status

Overall, we are currently working to better understand the relationships between network flows and how they can be exploited for improved performance. For example, we need to expand the range of log data we analyze and need to investigate the extent of host-to-cluster and cluster-to-host relationships.

Specifically, we are looking to better define the approach of an active network layer and evaluate its impact on both a host-to-host and cluster-to-cluster basis. There are outstanding issues of how this active network layer interacts with transport layer control, specifically how it handles or does not handle congestion control. As discussed in the next section, many mechanisms have been proposed to schedule packet transmission for congestion control, but have not investigated the opportunities of combining traffic to common hosts or clusters of hosts. Our active network layer could also be used to schedule transmissions of data under a particular rate control, but it may then need to notify the transport layer if a requested deadline cannot be met.

Other issues include adjusting the MTU of packets received from the transport layer so that space is available to aggregate these large packets with small packets. On the other hand, the transport layer should not need to deliver packets smaller than the MTU if there is no concurrent traffic.

8 Related Work

Previous work has examined techniques for exploiting relationships at a number of layers in the protocol hierarchy. In the following we look at three broad categories of how the techniques have been used at different layers, the types of relationships they exploit, and how they compare with our approach.

8.1 Shared or Centralized State Information

Much previous work has looked at shared or centralized state, primarily for purposes of congestion control. Applications, such as Web browsers, create multiple network flows to the same host for parallel retrieval of objects. Work on Ensemble-TCP [6] and shared TCP control blocks [26] are ways for these multiple TCP connections to share network information and better inform the TCP congestion control mechanism avoiding slow-start. This approach to congestion control is been implemented as part of the Linux kernel [20]. This technique is good for concurrent TCP flows and is also useful for sequential flows if the shared information is retained. This approach is limited to traffic of one (the most prevalent) transport protocol and does not reduce the number of transmitted packets.

Another approach to sharing is centralized scheduling of flows and packets. Work on the Congestion Manager (CM) [2] and the Internet Traffic Manager (ITM) [15, 5, 11] are examples of this approach where a manager schedules the transmission of packets taking into account congestion and QoS concerns. Several other studies discussed about sharing information among a cluster of hosts. In Ott and Mayer-Patel's coordination mechanism [17], they used an aggregate point (AP) on each cluster and inserted a coordination protocol (CP) layer between the IP and Transport layers. An AP calculates network conditions based on all flows passing through it and conveys the information to end hosts by CP. Pradhan et al. proposed an Aggregated TCP (ATCP) architecture [19], in which one TCP connection is segmented into two and joined by a local ATCP router on the sender's side. As a transparent TCP connection proxy, the ATCP router controls transmission rate on both sides based on information got from all connections going through it. In [21], Savage et al. introduced an overlay network called "Detour" and each node in "Detour" can aggregate traffic from its local hosts over tunnels (TCP connections).

8.2 Aggregation and Multiplexing

Another class of work has looked at aggregating traffic above the transport layer. An approach to aggregate traffic at the application layer is to multiplex data streams on top of a TCP connection. HTTP/1.1 [7] is an object-wise multiplexing scheme, which uses a persistent TCP connection to fetch multiple objects. Another approach to this same problem is to bundle multiple objects in one response [27]. Gilbert and Brodersen used layer-wise and byte-wise multiplexing schemes between a proxy server and a client [10]. SCP [23] and SMUX [9] are two general-purpose session control protocols that multiplex data from applications on one TCP connection.

The Stream Control Transmission Protocol (SCTP) [24] has multi-streams support, which permits bundling of more than one user message into a single SCTP packet, although SCTP can introduce a small delay as it tries to bundle. Users may disable bundling (like the PUSH flag is used in TCP) in order to avoid any delay. The use of bundling is similar to the approach we propose, but the protocol does not allow the application to set deadlines thus forcing applications with time dependencies to disable the mechanism.

8.3 Prediction

A third approach to using relationships between flows and packets is to perform work in anticipation of future work. At the transport layer, T/TCP [3] is one proposal to combine the TCP SYN and initial payload packet of a TCP connection setup therefore avoiding a round-trip between sender and receiver. T/TCP was proposed for transactions, but currently only FreeBSD has implemented it and its usage is still under experimental stage. Linux does not have plans for the implementation due to T/TCP's potential security problems [12]. In terms of number of transmitted frames, it is possible to emulate the transmission of multiple transactions by using the soft send feature to piggyback the creation of future TCP connections on current data transmissions.

Prediction is more commonly done at the application layer. Krishnamurthy et al. proposed a DNS-enabled Web approach that uses DNS messages to piggyback Web content in anticipation of future use [13]. In previous work, we have examined reducing DNS requests by piggybacking predicted future DNS responses [22].

9 Summary

The results of this work show a significant number of relationships exist among the network flows between pairs of hosts and host clusters. 40-50% of network flows between two hosts occurred within 10 seconds of a previous network flow between the same hosts while over 75% of flows exhibited this same relationship between two clusters.

Based on these relationships and on the fact that many packets delivered on the Internet are not full, we propose the idea of an active network layer that schedules the delivery of packets according to deadlines set by the application and transport layers while aggregating less-than-full packets as allowed by the MTU of the physical network. The availability of this network layer also allows the possibility of piggybacking duplicates of important packets on an as-available basis in order to improve the reliability of applications. We are currently working to more completely define and evaluate this approach on a host-to-host and cluster-to-cluster basis.

References

[1] Argus - IP network auditing facility.
`http://www.qosient.com/argus`.

[2] H. Balakrishnan, H. S. Rahul, and S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *Proceedings of the ACM SIGCOMM 1999 Conference*, pages 175–187. ACM, September 1999.

[3] R. Braden. T/TCP – TCP Extensions for Transactions Functional Specification, July 1994. RFC 1644.

[4] D.D. Clark and D.L. Tennenhouse. Architectural Considerations for a New Generation of Protocols. *ACM Computer Communication Review*, 20(4):200–208, September 1990.

[5] Gali Diamant, Leonid Veytser, Ibrahim Matta, Azer Bestavros, Mina Guirguis, Liang Guo, Yuting Zhang, and Sean Chen. itmBench: Generalized API for Internet Traffic Managers. Technical Report BU-CS-2003-032, CS Department, Boston University, Boston, MA 02215, December 2003.

[6] L. Eggert, J. Heidemann, and J. Touch. Effects of Ensemble-TCP. *ACM Computer Communication Review*, 30(1):15–29, January 2000.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, June 1999. RFC 2616.

[8] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-level traffic measurements from the Sprint IP backbone. *IEEE Network*, 2003.

[9] Jim Gettys and H. F. Nielsen. SMUX Protocol Specification, July 1998. Work In Progress (W3C Working Draft WD-mux-19980710).

[10] J. Gilbert and R. Brodersen. Globally Progressive Interactive Web Delivery. In *Proceedings of the IEEE Infocom 1999 Conference*. IEEE, March 1999.

[11] Liang Guo and Ibrahim Matta. Differentiated Control of Web Traffic: A Numerical Analysis. In *Proceedings of SPIE ITCOM'2002: Scalability and Traffic Control in IP Networks*, Boston, MA, August 2002.

[12] Charles M. Hannum. Security Problems Associated With T/TCP, September 1996.
`http://tcp-impl.grc.nasa.gov/tcp-impl/list/archive/1292.html`.

[13] Balachander Krishnamurthy, Richard Liston, and Michael Rabinovich. DEW: DNS-enhanced web for faster content delivery. In *Proceedings of the Twelfth International World Wide Web Conference*, Budapest, Hungary, May 2003.

[14] Balachander Krishnamurthy and Jia Wang. On network-aware clustering of web clients. In *Proceedings of the ACM SIGCOMM '00 Conference*, Stockholm, Sweden, August 2000.

[15] Ibrahim Matta and Azer Bestavros. QoS Controllers for the Internet. In *Proceedings of the NSF Workshop on Information Technology*, Cairo, Egypt, March 2000.

[16] J. Nagle. Congestion Control in IP/TCP internetworks, January 1984. RFC 896.

[17] D. Ott and K. Mayer-Patel. Transport-level Protocol Coordination in Cluster-to-Cluster Applications. In *Proceedings of 2002 USENIX Annual Technical Conference*, pages 147–159, June 2002.

[18] C. Perkins. IP Encapsulation within IP, October 1996. IETF RFC 2003.

[19] P. Pradhan, T. Chiueh, and A. Neogi. Aggregate TCP congestion control using multiple network probing. In *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS2000)*, April 2000.

[20] Pasi Sarolahti and Alexey Kuznetsov. Congestion Control in Linux TCP. In *Proceedings of 2002 USENIX Annual Technical Conference, Freenix Track*, pages 49–62, Monterey, CA, June 2002.

[21] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: Informed internet routing and transport. *IEEE Micro*, 19(1):50–59, January 1999.

[22] Hao Shang and Craig E. Wills. Using Related Domain Names to Improve DNS Performance. Technical Report WPI-CS-TR-03-35, Worcester Polytechnic Institute, December 2003.

[23] S. Spero. Session Control Protocol, Version 1.1. `http://www.w3.org/Protocols/HTTP-NG/http-ng-scp.html`.

[24] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, K. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol, October 2000. IETF RFC 2960.

[25] K. Thompson, G. J. Miller, and R. Wilder. Wide-Area Internet Traffic Patterns and Characteristics. *IEEE Network*, 11:10–23, November 1997.

[26] J. Touch. TCP Control Block Interdependence, April 1997. RFC 2140.

[27] Craig E. Wills, Gregory Trott, and Mikhail Mikhailov. Using bundles for web content delivery. *Computer Networks*, 42(6):797–817, August 2003.