Methodology For Developing Empirical Models of TCP-Based Applications*

(Extended Abstract)

F. Hernández Campos,[1] *K. Jeffay*, *F.D. Smith*
Department of Computer Science

S. *Marron*, *A. Nobel*
Department of Statistics

University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175 USA
http://www.cs.unc.edu/Research/dirt

Abstract: *We report on a large-scale empirical study to create application-level models for TCP traffic generation in simulations and network test-beds. A novel aspect of the study is the development of a method to construct empirical application-level models for arbitrary client/server (request/response) application-level protocols based on an analysis of unidirectional traces of only the TCP/IP headers in the traffic generated by the applications. By analyzing large collections of application-specific traces from a number of sources, we are producing empirical distributions to populate stochastic models of HTTP, FTP, and SMTP. However, combined, these applications represent less than 50% of the observed TCP traffic on our campus. We are also investigating the use of cluster analysis to identify sets of statistically homogenous TCP connection traces. Once these clusters have been identified we can then model each cluster by fitting stochastic models to the set of TCP connections identified as belonging to that cluster.*

1. Background and Motivation

A critical component of Internet simulations and test-bed measurements is the generation of synthetic traffic. Floyd and Paxson [12] provide an excellent analysis of the issues and pitfalls, especially traffic-generating models that are based on empirical data that has already been "shaped" by network influences. They conclude, "… if we take care to use traffic traces to characterize *source behavior*, rather than packet-level behavior, we can use the source-level descriptions in simulations to synthesize plausible traffic." In particular, for TCP-based applications, TCP's end-to-end congestion control (perhaps influenced by router-based mechanisms such as RED packet drops or ECN markings) shapes the low-level packet-by-packet traffic processes. Thus the generation of TCP traffic must be accomplished by using application-dependent but network-independent traffic sources layered over (real or simulated) TCP implementations. Two important pre-Web measurement efforts that produced application-dependent traffic models (Telnet, FTP, NNTP, SMTP), were conducted by Danzig *et al.*, [5, 10, 11], and by Paxson [15]. Web traffic generators in use today are usually based on data from two seminal measurement projects that focused on capturing web-browsing source behaviors: the Mah [13], and Barford, Crovella, *et al.*, [1, 3, 8, 9] studies. Traffic generators based on these sources have been built into the widely used *ns* network simulator [4] and have also been used to generate traffic in laboratory test-bed networks [2, 6].

Constructing traffic generators for TCP applications depends ultimately on the availability of high-quality Internet measurement data from a variety of sites that can be used to obtain characteristics of source (application-level) behavior. Our research is concerned with capturing and analyzing large-scale collections of Internet traces of TCP/IP protocol headers to create contemporary source-level models for traffic generation. We are further concerned with doing this with low-cost, low-overhead methods that can be used at multiple sites as an ongoing effort to create up-to-date models because Internet applications (and their use) continue to evolve rapidly.

We are currently working with traces of TCP/IP packet headers from two sources – (1) a collection we obtained by placing a network monitor on the Gigabit Ethernet link connecting the University of North Carolina at Chapel Hill (UNC) campus network to the Internet, and (2) a sample of traces taken from the collection maintained by NLANR/MOAT [18]. The UNC collection consists of traces taken in September 1999 (42 one-hour traces taken at six intervals on each of seven days), October 2000 (42 one-hour traces taken at six intervals on each of seven days), and April 2001 (21 four-hour traces taken at three intervals on each of seven days). The aggregate size of these TCP/IP headers is about 500GB. The NLANR/MOAT traces are from a number of sites and have been collected regularly over several years. Most of these traces only cover 90-second intervals and present problems in trying to capture very large data objects. We have, however, recently obtained traces from the NLANR/MOAT Auckland-IV collection [19] that are 24-hour traces taken at the University of Auckland by the WAND research group [20] during April 2001.

---

[1] Student author.

* This work supported in parts by grants from the National Science Foundation (grants CDA-9624662, ITR-0082870, and ITR-0082866), the Cisco, IBM, Intel, Sun, Cabletron, and Aprisma corporations, and NCNI.

2. Application-Specific Models

In [16] we described a method for analyzing *unidirectional* traces of only TCP/IP headers in order to infer application-level characteristics of HTTP. We showed that detailed information about the sizes of HTTP requests and responses, user browsing behaviors, the structure of web pages, and the use of the HTTP protocol (*e.g.*, the relative use of HTTP 1.0 *v.* HTTP 1.1), can all be determined without ever examining HTTP headers. Moreover, tracing only TCP/IP headers has the distinct advantages of easily allowing us to respect users' privacy (by anonymizing source and destination addresses in our traces) and reducing the complexity and storage overhead required for capturing application-level protocol headers. In addition, unidirectional tracing further reduces the acquisition overhead and eliminates problems with buffering and synchronization encountered when tracing simultaneously on two network interfaces (as is required when tracing on a fiber link).

The details of the analysis are described in [16] so we do not repeat that description here or the accompanying discussion of the complications and limitations of the method. We have now generalized the analysis methods so that they can be applied to TCP connections used for arbitrary application-level protocols and have used them to begin constructing models for FTP and SMTP.

The terminology used in describing this generalization is based on arbitrarily calling the TCP end-point initiating the connection (sends SYN) a *client* and the end-point accepting the connection (sends SYN+ACK) a *server*. The analysis assumes the application using a TCP connection implements a protocol where the client and server take turns sending application-defined protocol data units (PDUs) to each other in one or more *exchanges*. An exchange is defined as a pair of PDUs, one sent in each direction. Either the client or the server may send the initial PDU after the connection is established and an incomplete exchange (one endpoint does not send a PDU) is allowed. For example, in HTTP, an exchange consists of an HTTP-request PDU sent from the client to the server and an HTTP-response PDU sent from the server to the client (more exchanges may follow if persistent connections are used). FTP-CONTROL connections follow a multi-exchange pattern of FTP command and response PDUs while an FTP-DATA connection is comprised of a single (incomplete) exchange where the client either sends or receives one PDU that is the file (or directory listing). SMTP connections typically have several exchanges of SMTP command and response PDUs, along with PDUs that are email messages.

Since our analysis method sees only the TCP segments flowing in one direction on the connection, it uses changes in sequence numbers in those segments to compute PDU sizes flowing in the direction traced and changes in ACK values to infer PDU sizes flowing in the opposite direction (not traced). Fundamental to the analysis is the assumption of alternating PDU flows (one in each direction per exchange). There will be an alternating pattern of advances in the ACK values followed by advances in the data sequence values (or vice versa). This observation is used to construct a rule for inferring the beginning and ending TCP segments of a PDU and the boundary between exchanges. Put another way, an advance in the data sequence numbers marks the end of a PDU flowing in the direction opposite the traced direction and an advance in the ACK sequence number marks the end of a PDU flowing in the direction of the trace. Of course, other events such as FIN or Reset can mark ends also. The (rare) application protocols that overlap rather than alternate exchanges between client and server are not handled correctly by this method.

As an example of the analysis, suppose that after the SYN (or SYN+ACK), the ACK sequence numbers in subsequent TCP segments advance. We infer that data-carrying segments for a PDU flowed in the direction opposite the one captured in the unidirectional trace. As long as the ACK values advance but data sequence numbers do not, the amount the ACK values change is assumed to indicate the size of that PDU. When the data sequence numbers begin advancing, we infer that the end of the first PDU has been reached and a PDU flowing in the direction of the trace has begun. Its length is indicated by the amount the data sequence numbers advance before the next advances in the ACK values (marking the end of the exchange and the beginning of a new PDU flowing in the other direction for the next exchange) or the connection ends. Similarly, suppose that after the SYN (or SYN+ACK), the data sequence numbers advance but the ACK values have not. This shows that data-carrying segments for a PDU are flowing in the direction captured in the trace. As long as the data sequence numbers advance but ACK sequence numbers do not, the amount the data sequence numbers change is assumed to indicate the size of that PDU. When the ACK sequence numbers begin advancing, we infer that the end of the first PDU has been reached and a PDU flowing in the opposite direction of the trace has begun. Its length is indicated by the amount the ACK sequence numbers advance before the next advances in the data sequence numbers (marking the end of the exchange and the beginning of a new PDU flowing in the other direction for the next exchange) or the connection ends. If both ACK and data sequence values advance in a single segment, their inferred meaning depends on the prior state of the connection.

This use of alternating advances in ACK or data sequence values to mark PDU exchanges is disturbed by segment reorderings in the network. In some cases, such as reordering of segments that carry only data or only an ACK, there is usually no problem since only the highest value seen is used. Reordering of segments carrying both data and ACKs presents problems since boundaries between PDUs and exchanges may be missed which can result in overstating or understating PDU sizes. For this reason, cases of segment re-ordering that involve both data and ACK values require special case analysis. In addition, as described in [16] for HTTP, other application-level phenomena have the potential to limit our ability to precisely characterize an application's behavior.

We have applied this analysis method to TCP connections for HTTP, FTP, and SMTP found in our 2000 and 2001 traces from the UNC Internet link. For HTTP and FTP-DATA connections, the sizes of transferred objects (files)

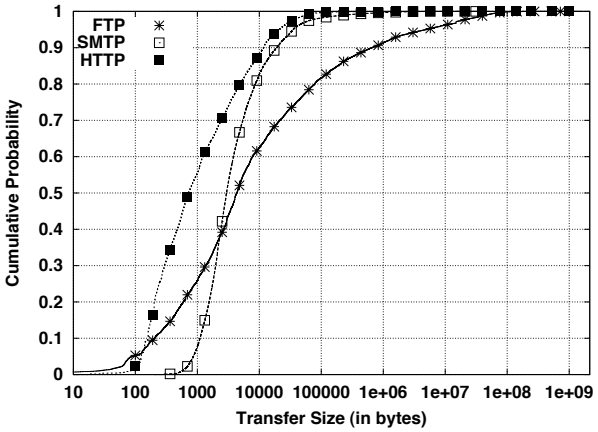

Figure 1: Cumulative distribution of HTTP response sizes, FTP-DATA file sizes, and SMTP email message sizes.

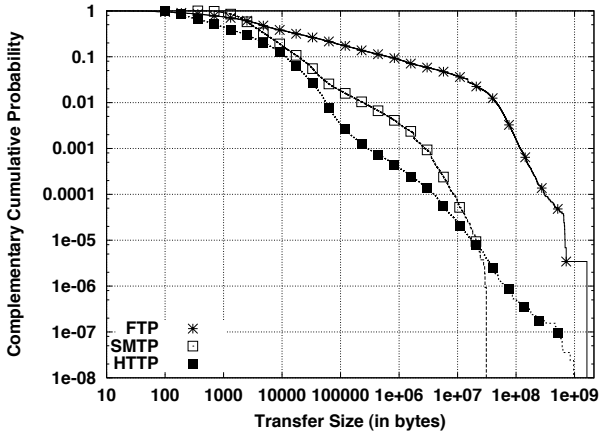

Figure 2: Complementary cumulative distribution of HTTP response sizes, FTP-DATA file sizes and SMTP email message sizes.

can be obtained directly from the PDU sizes computed by the analysis program. In the case of SMTP connections it is possible to apply a simple heuristic to the resulting PDU sizes to obtain a distribution of email message sizes. In the SMTP protocol, normally when the client sends a command PDU consisting of 6 bytes ("DATA\n\r"), the next PDU it sends is the email message itself. However, other commands may also be 6 bytes long and the client may send another command PDU following "DATA\n\r" instead of an email message if the server does not accept the DATA command. We have determined empirically that a simple size test can be used to construct the following rule for identifying an email message: if the next client PDU following a client PDU of 6 bytes is larger than 200 bytes, it is (with very high probability) an email message.

At a high level, these application protocols have similar functions in that they are designed to move application-defined data elements (HTTP = web object, FTP-DATA = file, SMTP = email message) from one location to another. It is interesting to consider how the sizes of these data elements differ from one application to another. Figure 1 gives the cumulative distribution of data sizes for HTTP, FTP-DATA, and SMTP from the UNC traces. Figure 1 clearly shows the differences among the size distributions for application-defined data. For example, HTTP has a much higher proportion of small objects, especially those smaller than 2K bytes. As we might expect, SMTP message sizes tend to be concentrated in a much narrower range (1000 – 10000 bytes) than HTTP or FTP-DATA. Figure 2 shows the same data plotted as a complementary CDF in order to emphasize the comparison among the distribution tails. Again we see clear distinctions among the three. Perhaps the most interesting observation is that SMPT email message sizes have heavy-tail characteristics similar to HTTP and FTP-DATA. This probably reflects the common use of email attachments as an alternative to HTTP or FTP for moving large objects.

By analyzing large collections of application-specific traces from a number of sources, we are producing empirical distributions to populate stochastic models of HTTP, FTP, and SMTP. For example, an application-level model of SMTP requires the following: a distribution of inter-arrival times for SMTP connections, a distribution of the number of command-response exchanges per connection, distributions of command and response sizes, a distribution of the number of email messages per connection, and a distribution of email message sizes. The next step in this process is to fit analytic models to the empirical distributions constructed for these three applications.

3. Application Class Models

While it is relatively straightforward to create models of specific applications based on widely-known protocols such as HTTP, FTP, SMTP, this approach does not scale when our goal is to have models for all the 90-95% of Internet traffic that is TCP [7, 14, 17]. For example, consider the following results from our 2001 traces. These traces contain the TCP/IP headers from all packets flowing to UNC from the Internet during 84 hours of tracing spread over seven days. In this time, 5.35 billion packets entered the university network carrying 1,837 billion bytes of data. Of this, 90.4% of the bytes (91.4% of the packets) were TCP. The four largest TCP-based applications (as measured in bytes of inbound traffic) that could be easily identified by "well-known" port numbers (< 1024) were HTTP (port 80), FTP-DATA (port 20), NNTP (port 119), and SMTP (port 25). These four accounted for less than half of the total TCP bytes (HTTP: 39.7%, FTP-DATA: 5.3%, NNTP: 1.4%, and SMTP: 1.1%). The remainder of the TCP bytes (52.5%) were carried on TCP connections using a very large number of TCP port pairs most of which are listed in the IANA protocol port assignments as "unassigned" (connections using pairs of unassigned ports carried 30% of the TCP bytes).

Most of the TCP-based applications that don't use well-known ports individually carry far less than 1% of the TCP bytes (there are some quite notable exceptions such as Napster and Gnutella). There are, however, so many different port pairings that, in aggregate, they account for approximately as much inbound traffic as HTTP, FTP-DATA,

NNTP and SMTP combined. Clearly, trying to model the traffic characteristics of each application individually is a nearly impossible task.

We are addressing this problem by creating a framework of *application class models* where each class is distinguished by a common paradigm of how it is a source of network traffic. For each such class, we will create a single stochastic model that reflects the aggregated characteristics of all TCP-based applications that are members of that class. We are using statistical cluster analysis techniques to identify a set of such classes based on traces from several Internet links. If successful, we should be able to represent a very large number of applications with a relatively small number of models (perhaps 10-20). Given this set of class-based models and a desired proportion of each class, one can generate the application-level data characteristics corresponding to an arbitrary mix of applications.

Our cluster analysis approach is based on computing for each individual TCP connection seen in a trace a "feature vector" of values that characterizes it. This approach depends on using the analysis methods described above for processing the TCP/IP headers from a unidirectional trace of the connection to identify exchanges of application-level protocol data units in the same way we did for HTTP, FTP and SMTP. Once the sizes and boundaries of the exchanges in a connection are identified we can compute uni- and multivariate statistics about the exchanged application-defined PDUs to form a feature vector for that TCP connection. For example, our initial feature vector includes:

- Total bytes and number of exchanges along with the maximum, minimum, median, variance, total variation, maximum first difference, auto-correlation, and homogeneity index of the sizes of PDUs in each direction,
- Covariance, cross-correlation, and directionality index among the data-unit sizes in all exchanges considered as a sample path combining both directions, and
- Relative traffic intensity in each direction of the exchange.

A number of statistical tools are available that automatically divide sets of feature vectors into clusters based on minimizing measures of "distances" between vectors within a cluster while maximizing distances between clusters. Once the clusters have been identified, we can then model each cluster by fitting stochastic models to the set of TCP connections identified as belonging to that cluster. The critical research issue in completing this analysis will be determining those features that yield a useful set of statistically homogenous clusters of TCP traces.

4. References

[1] P. Barford, M. Crovella, *Generating Representative Web Workloads for Network and Server Performance Evaluation*, Proc. ACM SIGMETRICS '98, 1998, pp. 151-160.

[2] P. Barford and M. E. Crovella, *A Performance Evaluation of HyperText Transfer Protocols*, Proceedings of ACM SIGMETRICS '99, May 1999, pp. 188-197.

[3] P. Barford, A. Bestavros, A. Bradley, and M. E. Crovella, *Changes in Web Client Access Patterns: Characteristics and Caching Implications*, World Wide Web, Special Issue on Characterization and Performance Evaluation, Vol. 2, 1999, pp. 15-28.

[4] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, H. Yu, *Advances in Network Simulation*, IEEE Computer, vol. 33 no. 5, May 2000, pp. 59-67.

[5] R. Caceres, P. Danzig, S. Jamin, D. Mitzel, *Characteristics of Wide-Area TCP/IP Conversations*, Proceedings of ACM SIGCOMM '91, pp. 101-112.

[6] M. Christiansen, K. Jeffay, D. Ott, F. D. Smith, *Tuning RED for Web Traffic*, Proceedings of ACM SIGCOMM 2000, September 2000, pp. 139-150.

[7] K Claffy, G. Miller, K. Thompson. *The nature of the beast: recent traffic measurements from an Internet backbone*, Proccedings of INET '98, (*http://www.isoc.org/inet98/proceedings/6g/6g_3.htm*).

[8] M. Crovella,, A. Bestavros, *Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes*, IEEE/ACM Transactions on Networking, vol. 5, no. 6, December 1997, pp. 835-846.

[9] C. R. Cunha, A. Bestavros, M. Crovella, *Characteristics of WWW Client-based Traces*, Tech. Report TR-95-010, Boston University Computer Science Dept, June 1995.

[10] P. Danzig, S. Jamin, *tcplib: A Library of TCP Internetwork Traffic Characteristics*, USC Technical Report USC-CS-91-495, 1991.

[11] P. Danzig, S. Jamin, R. Caceres, D. Mitzel, D. Estrin, *An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations*, Internetworking: Research and Experience, vol. 3, no. 1, 1992, pp. 1-26.

[12] S. Floyd, V. Paxson, *Difficulties in Simulating the Internet*, IEEE/ACM Transactions on Networking (to appear). Available at *http://www.aciri.org/vern/papers/ sim-difficulty.TON.2001.pdf*.

[13] B. Mah. *An Empirical Model of HTTP Network Traffic*, Proceedings of IEEE INFOCOM '97, April 1997. (An extended version is at *http://www.ca.sandia.gov/~bmah/ Software/HttpModel/*)

[14] S. McCreary, K. Claffy, *Trends in Wide Area IP Traffic Patterns: A View from Ames Internet Exchange*, CAIDA Tech. (*http://www.caida.org/outreach/papers/AIX0005/*).

[15] V. Paxson, *Empirically Derived Analytic Models of Wide-Area TCP Connections*, IEEE/ACM Transactions on Networking, vol. 2, no. 4, August 1994, pp. 316-336.

[16] F.D. Smith, F. Hernandez Campos, K. Jeffay, *What TCP/IP Protocol Headers Can Tell Us About the Web,* Proc. ACM SIGMETRICS '01, June 2001, pp. 245-256.

[17] K. Thompson, G. Miller, R. Wilder, *Wide-Area Internet Traffic Patterns and Characteristics*, IEEE Network, vol. 11 no. 6, November/December 1997.

[18] *http://moat.nlanr/net/Traces/*

[19] *http://moat.nlanr.net/Traces/Kiwitraces/auck4.html.*

[20] *http://wand.cs.waikato.ac.nz/*