Internet Traffic Data

William S. Cleveland and Don X. Sun
Statistics and Data Mining Research, Bell Labs
wsc@bell-labs.com

Internet engineering and management depend on an understanding of the characteristics of network traffic. Statistical models are needed that can generate traffic that mimics closely the observed behavior on live Internet wires. Models can be used on their own for some tasks and combined with network simulators for others. But the challenge of model development is immense. Internet traffic data are ferocious. Their statistical properties are complex, databases are very large, Internet network topology is vast, and the engineering mechanism is intricate and introduces feedback into the traffic. Packet header collection and organization of the headers into connection flows yields data rich in information about traffic characteristics and serves as an excellent framework for modeling. Many existing statistical tools and models — especially those for time series, point processes, and marked point process — can be used to describe and model the statistical characteristics, taking into account the structure of the Internet, but new tools and models are needed.

1. The Internet

Internet traffic data are exciting because they measure an intricate, fast-growing network connecting up the world, transforming culture, politics, and business. And a deep understanding of Internet traffic can contribute substantially to network performance monitoring, equipment planning, quality of service, security, and the engineering of Internet communications technology. Two ingredients are required for this understanding: frameworks for traffic measurement that produce data bearing on the Internet issues, and statistical models for the data. Measurement has received expert, effective attention. A good start has been made in modeling but much more can be done. This first section contains a description of some of the fundamentals of Internet communication, serving as a basis for the discussions of measurement and modeling in later sections.

Packets and TCP/IP

Each Internet communication consists of a transfer of information from one computer to another; examples are the downloading of a Web page and the sending of an email message. When a file is transferred, it is not sent across the Internet as a continuous block of bits. Rather the file is broken up into pieces called *packets*, and each packet is sent individually. Many different *protocols* collectively carry out the transfer. An Internet protocol is simply a set of rules for communication between computers. The two core protocols are TCP, the Transmission Control Protocol and IP, the Internet Protocol.

TCP runs on both computers. It breaks up a file to be transferred into packets, sends them out from the source headed for the destination, receives them at the destination, and reassembles them into their proper order. A typical packet size is 1460 bytes. So if the abstract of this paper were sent by email as a text file, about 1000 bytes, it would fit into a single packet. If the whole paper were sent as a postscript file, about 250 kilobytes, it would take 170 packets.

TCP does the transfer by establishing a *connection* between the computers. The connection is not a physical path; rather, it is simply TCP software executing on the computers in a coordinated fashion, with each aware that it is working with the other. The connection continues until both sides agree that it is over, or until one side fails to hear from the other for a specified amount of time. Packets go back and forth between the two computers. For a packet, the *source* computer is the one that sends it out, the *destination* computer is the one that receives it.

IP is in charge of routing TCP's packets across the Internet. Each computer has an IP address, a unique 32 bit number. Often, the number is displayed by dividing the sequence of bits into four 8-bit fields, writing each field as a decimal number between 0 and 255, and then displaying the four numbers separated by dots. One example is 135.104.13.160. The number of possible IP addresses is 2^{32} which is about 4 billion, but just to show how fast the Internet is growing, plans have been made to switch to 128 bit addresses.

An IP header is added to each packet. The header includes, among other things, the source IP address, the destination IP address, and the packet size. *Routers* are Internet devices that get the packet to its destination. The packet moves from one router to the next, each reading the destination address in the header, and looking in a table to find the router to which it should forward the packet. The source, the series of routers, and the destination from a path across the Internet. The packets of a single file transfer do not have to follow the same path; for example, if a path in use at the beginning of the transfer becomes unavailable, tables are updated by communication among the routers, and a new path is used.

A TCP header is also added to each packet. The header contains information to control the connection and to reassemble packets. TCP also creates packets that contain only headers but no file information. Control is their only purpose and they are generated by both computers. For example, a connection is initiated by one computer transmitting a *synchronization*, or *SYN*, packet to the other. The control information in headers plays a role in a TCP feedback mechanism that is fundamental to the operation of the Internet and has a major effect on traffic. At the onset of a connection, TCP sends packets at a conservative rate and then progressively increases the rate. If a router receives packets faster than it can forward them, it places the overflow in a buffer; if the buffer overflows, packets are dropped. If packets are dropped, TCP decreases the transmission rate, retransmits the dropped packets, and then progressively increases the rate again.

Each TCP connection is set up at the request of an Internet application. For example, the application HTTP (Hypertext Transfer Protocol) transfers a World Wide Web page from a *server* computer to a *client* computer. SMTP (Simple Mail Transfer Protocol) sends email. Telnet enables logging on to remote computers, transferring input on the client to the remote computer, and transferring responses back. FTP (File Transfer Protocol) transfers files between local and remote computers. A single invocation of an application can result in many TCP connections. For example, a Web page request using the most common version of HTTP results in a connection to transfer the linked file, and a connection for the transfer of each embedded image file in the linked file.

Network Topology and End-to-End Connections

The Internet is a network of networks. Here is a simplified but useful view of it. There are end-points, end-networks, and a core. The end-networks are the networks of companies, universities, and other organizations. The end-points are individual home users, typically with a single connected computer, although little end-networks are forming in homes too. The core is a collection of ISPs, Internet Service Providers such as AOL-Time-Warner, AT&T, BBN, and MCI. Each end-point or end-network connects to an ISP, and the ISPs have interconnection points.

Suppose a home user downloads a Web page from a large university computer. An *end-to-end* connection is established between client and server. The first packet, the SYN, starts out at the client, and goes over a wire to the ISP. Today this wire might be a phone line or a television cable and can range from about 28.8 kilobits/sec to 2 megabits/sec. These transmission rates refer to the speed at which the two devices at the two ends of the wire can send and receive bits; once the bits get on the wire they travel at the speed of light. Then the packet moves through the ISP's network, perhaps transfers to another ISP, but eventually hits the ISP providing service to the university. Today ISP backbone wires, which are fiber, have transmission rates of up to 10 gigabits/sec. Then the packet enters the university network and travels through the university's wires and network devices until it is routed to the university computer with the Web page. Transmission speeds on the university network might vary over the different links from 10 megabits/sec to 1 gigabit/sec. The travel time of the packet depends on the distance of the university from the home, the speed of the link on the path, and the congestion encountered along the way, but tens of milliseconds to the low hundreds would be common. When the server receives the SYN it sends a control packet back to the client, acknowledging the SYN; the client then sends a control packet back to the server, acknowledging the acknowledgement, and the real job begins, transferring information.

2. Flow Measurement

One effective framework for traffic measurement is TCP/IP packet header collection, and organization of the headers into TCP connection *flows*. The framework has been in place throughout much of the short history of the Internet, and important fundamental work has arisen from it (Caceres, Danzig, Jamin, and Mitzel 1991; Mogul 1992; Claffy, Braun, and Polyzos 1995; Paxson 1997a). This section discusses the framework and its uses.

Connection Flows

Consider a wire carrying Internet traffic. An example is the wire that connects a Bell Labs Research network of about 3000 machines to the rest of the Internet. We collect headers on this wire. There are two directions: packets coming into the Bell Labs network from computers outside, and packets traveling outside from computers inside. Packets pass by one by one, and packets from different TCP connections are superposed in the sense that at any one time there can be many TCP connections in progress, so we might see a packet from one connection, and then a packet from a different connection, and then a packet from the first. We read and copy the TCP/IP headers of each packet and add a timestamp, the arrival time of the packet. Then an algorithm is used to disentangle the packets and form individual TCP connection flows. The headers for each connection flow are stored together in the database in order of the time of arrival.

End-to-End Characteristics

TCP connection flows provide a large amount of information. Each flow is an end-to-end connection traversing the Internet. For the Bell Labs wire, one end is a computer inside Bell Labs and the other is a computer outside. The TCP/IP headers contain the IP addresses of the two computers, so we know their location in the vast Internet topology. Thus flows can be used to study network-wide characteristics. Here is one example involving HTTP. For this Bell Labs wire, there are no incoming Web page requests, just outgoing; traffic to and from Bell Labs Web servers passes over other wires. On the collection wire, the client computers for HTTP, the ones that request pages, are inside, and the server computers, the ones that supply pages, are outside. For each HTTP flow we can compute its duration, the time of the last packet minus the time of the first packet, and we can compute the size of the transferred file. Then we can divide size by time to get throughput in bytes/sec. This throughput variable is one measure of the quality of the Web transfers that are requested by users in the Bell Labs network. Bigger is better. We can study this quality metric and, using the destination IP address, determine how it varies topologically across the Internet, or adding the timestamp information, how it varies with time of day and topology. Of course, we can do the same for other applications as well.

Aggregate Traffic on the Wire

A TCP connection flow database also provides information about the traffic on the wire. The TCP/IP headers have the size of each packet in bytes, so together with the timestamps, we have the aggregated packet process: the arrival times and sizes of all packets. Studying aggregates is important because the devices at each end must handle packets, in time order, and the performance of the devices depends on the packet inter-arrival times and the packet sizes. Forming the aggregate of all packets from the flows takes us back to the packet information in its original state: packets in time order. But storage by connection flow is still important because we often study sub-aggregate traffic: time-ordered packets from a subset of the flows. For example, each flow results from an application such as HTTP, FTP, SMTP, or Telnet requesting a connection and transfer of information; it is important to study aggregate traffic by application because the packet processes for different applications are different. We can also study derived processes formed from any sub-aggregate. A common one is byte counts; time is divided up into intervals of equal length, and the number of bytes of packets arriving in each interval is computed.

3. Modeling

Internet traffic data are ferocious. Their statistical properties are complex and databases are very large. The protocols are complex and introduce feedback into the traffic system. Added to this is the vastness of the Internet network topology. This challenges analysis and modeling. Most Internet traffic data can be thought of as time data: a point process, a marked point process, or a time series. The start times of TCP connection flows for HTTP on an Internet wire are a point process. If we add to each of these start times the file size downloaded from the server to the client, the result is a marked point process. Byte counts of aggregate traffic summed over equally spaced intervals are a time series. Modeling Internet traffic data will require new approaches, new tools, and new models for time data. In this section we look back at the current body of literature on Internet traffic studies to formulate issues for this new work. Other such discussions may be found in two interesting papers of Willinger and Paxson (1997) and Floyd and Paxson (1999).

HTTP Start Times

We will begin with an example of traffic data, and their analysis, to serve as a backdrop for the discussion (Cleveland, Lin, and Sun 2000a). The data are HTTP start times on the Bell Labs wire described above. When a Bell Labs user clicks on a link, the linked file is downloaded from the server outside to the client inside; the client's HTTP sees the names of embedded image files and requests they be downloaded. For the most prevalent version of HTTP, each file is transferred by its own TCP connection. So a single click can open many TCP connection flows. The SYN packet sent from the client that begins the connection travels through the Bell Labs network and arrives on the measurement wire. As part of the overall header collection we read and copy the TCP/IP headers on the SYN, and record the arrival time. The measured HTTP start time on the wire is a bit later than the actual start time on the client, because it takes time for the packet to travel from the client to the wire, but there is typically only a small delay inside the Bell Labs network compared to the travel time outside.

The HTTP start times are a superposition point process. Each user generates an HTTP start time point process on the wire, and the aggregate start time process is the superposition of the user point processes. The data used in the study cover the period 11/18/98 to 7/10/99. There are 23,008,664 measured start times during this period. We organized the data into 15-minute blocks. Not every block during the measurement period appears due to monitor down time or low usage. We eliminated blocks for which the monitoring was not operational more than 5% of the full 15 minutes, blocks with fewer than 50 flows, and connections from certain hosts that developed problems. The result was 10704 blocks of start times. We assume that the point process of start times is stationary within a block.

The bottom panel of Figure 1, an *inter-arrival plot* (Cleveland, Lin, and Sun 2000a), displays the 2515 HTTP start times for one block, 7:45 a.m. to 8:00 a.m. on 12/11/98. Let s_k for k = 1 to 2515 be the start times, and let $t_k = s_k - s_{k-1}$ for k = 2 to 2515 be the inter-arrival times. On the plot, $\ell_k = \log_2(t_k)$ is graphed against s_k where $\log_2$ is the log base 2. The log on the vertical scale is vital because inter-arrivals can vary over 16 powers of 2, a factor of about 64000, and small intervals are as important as large ones; so the vertical scale provides the requisite resolution to see this variation. The horizontal scale, however, conveys arrivals and inter-arrivals on the original scale. The connection rate over this block in connections per sec, or c/s, is 2515/900 c/s = 2.8 c/s. The top panel of Figure 1 is an inter-arrival plot for the block 12:15 p.m. to 12:30 p.m. on the same day. The connection rate, 20.1 c/s, is greater than that for the bottom panel because there are more users browsing the Web from 12:15 p.m. to 12:30 p.m. than from 7:45 a.m. to 8:00 a.m.; in other words, the amount of superposition in the later period is greater.

Both panels show discreteness on the vertical scale, inter-arrivals piling up at a few values. This is a network effect, a small delay; each accumulation point is a time equal to the time it takes to process a packet in the network. For example, suppose two SYN packets are back to back, which happens a small fraction of the time. They arrive on the wire; each is timestamped and then read by the PC. The inter-arrival time is the time it takes to read the first packet, which is the packet size times the wire speed; at the time of collection the speed was 10 megabits/sec.

The different levels of superposition in the two panels of Figure 1 dramatically change the statistical properties of the inter-arrival times. In the bottom panel, the data form distinct, narrow vertical bands for which the ℓ_k range from about -8 $\log_2$sec to about -2 $\log_2$sec. The bands are bursts of connections caused by single clicks of individual users. The number of connections in such bursts can be large because the number of embedded files has a distribution with a long upper tail (Feldmann 1998; Barford and Crovella 1998). In the top panel of Figure 1, the bursty behavior has disappeared. Because the connection rate is much higher, the SYNs of more users intermingle, and the behavior of individual users is broken up. There is another difference between the two panels. The variance of the ℓ_k is less for the higher-rate start times than for the lower. For both cases, the ℓ_k sequence exhibits *long-range persistence*: a slowly decaying positive

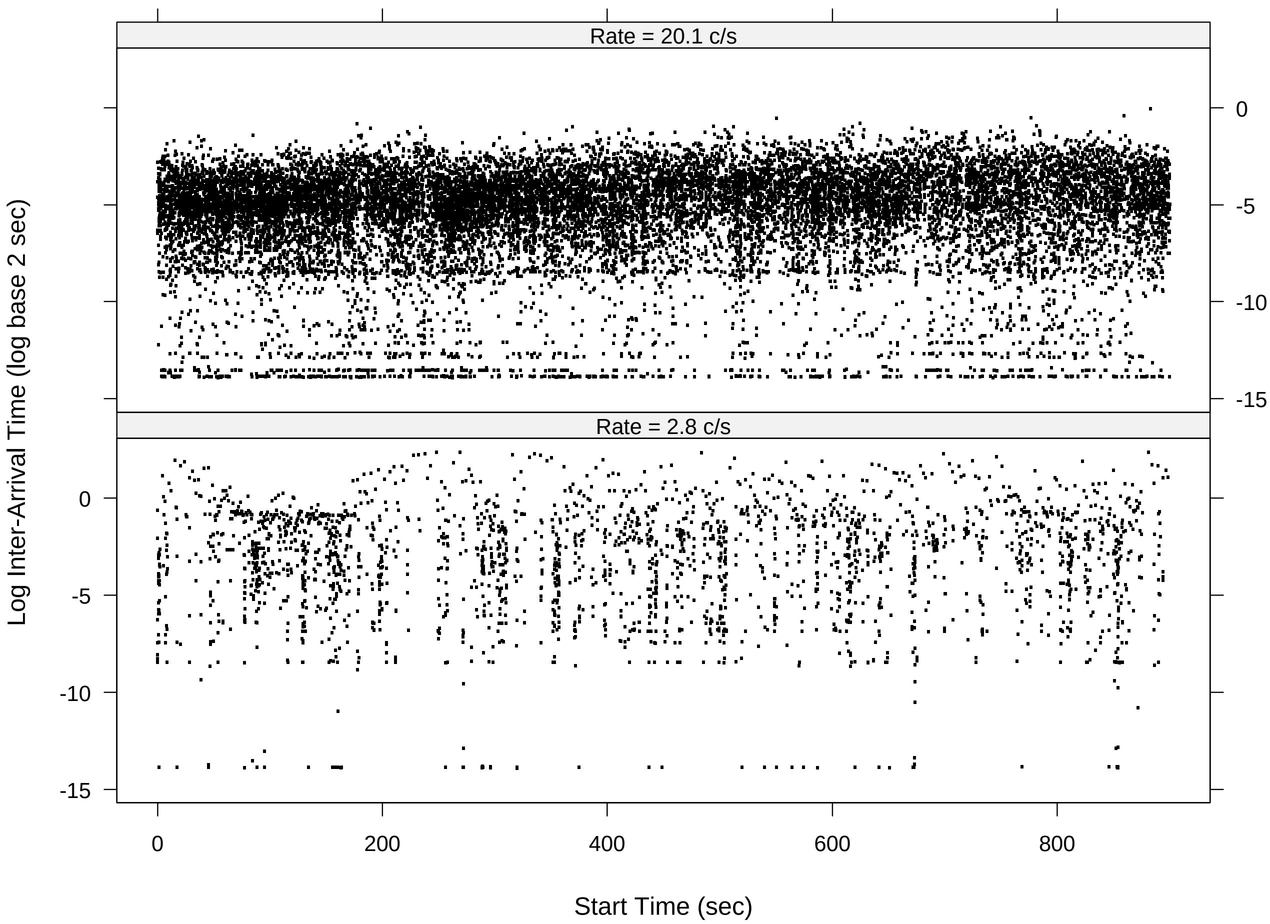

Figure 1: HTTP start times are displayed. Log base 2 inter-arrival time is graphed against start time for two 15 minute blocks of HTTP start times. The connection rate in the top panel is greater than in the bottom because there is more Web usage. The higher rate reduces the amount of autocorrelation of the log inter-arrival time sequence and makes the Weibull marginal distribution of the inter-arrival times closer to an exponential.

autocorrelation function. In the top panel, because of the increased superposition, the autocorrelation is much reduced compared with the bottom panel. The marginal empirical distribution of the t_k for each panel is well approximated by a Weibull (except for the discreteness), with a shape parameter λ less than 1, the shape for an exponential, but the value of λ for the bottom panel is smaller than for the top.

The analysis of the HTTP start times produced the following conclusions (Cleveland, Lin, and Sun 2000a). The marginal distribution of the t_k is approximately Weibull with λ less than 1, and as the connection rate increases, λ tends toward 1. The change in the distribution over the intervals is large; the estimate of λ ranges from about 0.4 to 0.9. The autocorrelation of the ℓ_k is described by an exceedingly simple model with two parameters: white noise plus a long-range persistent time series. The two parameters are the variances of the two series. As the connection rate increases, the variance of the persistent series tends toward zero, and the ℓ_k tend toward independence. The change in the autocorrelation function over the intervals is large; the fraction of the variance of ℓ_k accounted for by the persistent series ranges from about 0.5 to 0.1.

Superposition

Aggregated Internet traffic is a superposition of traffic sources. It is vital to exploit superposition to uncover the characteristics of Internet traffic. In so doing, we exploit the fundamental structure of the traffic. We can operate mathematically, using the theory of superposition of point processes, marked point processes, and time series. We can operate empirically, studying the data and how it changes as the number of sources changes.

The notion of how we define a source for analysis purposes needs more thought and trial with data. For example, for the Bell Labs HTTP start times, we can take sources to be users. However, the Bell Labs network is actually a network of sub-networks. So we could take each source to be the start times of one sub-network. There is another method of approaching superposition that avoids explicit identification of sources. It is *rate superposition*: traffic rates are used as a measure of the number of traffic sources (Cleveland, Lin, and Sun 2000a; Cao, Cleveland, and Sun 2000). This method was employed in the analysis and modeling of the HTTP start times. A low base rate was selected, r_0, close to the minimum observed rate. Let k be a positive integer. The start times over a period of 15 minutes in an interval with rate kr_0 were taken to be the superposition of k independent start-time point processes, each with the statistical properties of the point process in an interval with the base rate. Theoretical results were derived based on the superposition theory of point processes. The results were compared with the empirical results, the analyses of the 10704 intervals of start times, and agreement was excellent. The conclusions about the HTTP start times just presented resulted from this combination of theory and empiricism.

Long-Range Persistence and Long Upper Tails

Long-range persistence is pervasive in Internet traffic data. But the pervasiveness had to be discovered (Leland, Taqqu, Willinger, and Wilson 1994; Paxson and Floyd 1995; Willinger, Taqqu, Leland, and Wilson 1995). Traffic models for voice traffic developed over the years to serve the telephone network did not apply as might have been hoped because voice traffic does not give rise to the same traffic characteristics as Internet data traffic, which is burstier. As we have seen, the HTTP start times reveal persistence.

The reason for the increased burstiness in some cases is a combination of superposition and distributions with very long tails (Crovella and Bestavros 1996; Willinger, Taqqu, Sherman, and Wilson 1997). Suppose we look at HTTP byte counts over equal-length intervals on an Internet wire carrying Web traffic. The counts for each interval result from the superposition of TCP connection flows, each transferring a file. So the byte counts in an interval are the sum of byte counts in the interval for a number of ongoing connections. The distribution of Web file sizes has an extremely long upper tail; in fact, the density falls off for large x like $x^{-\beta}$ where $1 < \beta < 2$, which means the size distribution has an infinite variance (Crovella and Bestavros 1996). So interspersed among the common-sized files will be immense files; their transfers raise the byte count level over a very long period, creating positive autocorrelation extending over a long range of lags.

The discovery of long-range persistence was an important piece of basic science for the analysis of network traffic, not just because it served to establish a fundamental characteristic of Internet data, but also because it served notice that the common practice of simply assuming a model for network traffic was defective and that looking at data was important. It was also an important discovery for the engineering of the Internet; the burstiness has required communication algorithms in some cases different from those that would be needed for much less bursty data (Erramilli, Narayan, and Willinger

1996) although not in all cases (Ryu and Elwalid 1996).

After the discovery of long-range persistence, a number of methods for characterizing the dependence were employed that were appropriate to self-similar processes and that came from the creative work of Mandelbrot (1968). Consider a stationary time series y_t for integer t. Form a new time series,

$$y_v^{(m)} = m^{-H} \sum_{k=1}^{m} y_{vm+k}$$

for integer v and positive integer m. Then y_t is self similar with parameter H if for all m the finite sample distributions of the $y_v^{(m)}$ are the same as those of y_t. Self-similar processes with $0.5 < H < 1$ are persistent processes, and the autocorrelation at lag k decays like $k^{2(H-1)}$.

But self-similarity in the strict sense is exceedingly restrictive and Internet data are not self-similar (Feldman, Gilbert, and Willinger 1998; Ribeiro, Riedi, Crouse, and Baraniuk 1999; Floyd and Paxson 1999). At very small time scales, hundreds of milliseconds and less, the behavior of Internet protocols can be dominant, and at time scales of tens of minutes and more, diurnal variation can be dominant. In between are the time scales of the persistent processes. Thus, there are various components, and some are persistent. Multifractal wavelet methods have become a widely-used tool to study the components (Riedi and Vehel 1997; Feldman, Gilbert, and Willinger 1998; Ribeiro, Riedi, Crouse, and Baraniuk 1999). Another approach is to build time domain models and include long-range persistent components as is done in other disciplines (Hosking 1981; Haslett and Raftery 1989). This latter approach was used in the analysis of the Bell Labs HTTP start times.

Nonstationarity

Nonstationarity is as pervasive in Internet traffic data as long-range persistence. (Cleveland, Lin, and Sun 2000a; Cao, Cleveland, and Sun 2000). However, nonstationarity has received much less attention. More attention is surely needed since the nonstationarity can be major. For example, for the Bell Labs HTTP start times, the autocorrelation function and univariate distributions changed substantially. The reason for the nonstationarity is quite simple — changing amounts of superposition. As the number of traffic sources increases and decreases, statistical properties can change. Byte counts on a wire are sums of byte counts of the individual sources, and the finite sample distributions of sums of random variables can change with the number of terms in the sums. HTTP start times on a wire are a superposition of the individual traffic sources, and the finite sample distributions of the inter-arrival times of superposed point processes can change with the number of superposed processes.

Accounting for Structure and Linking Up with Network Simulators

Willinger and Paxson (1997) argue effectively against black box modeling that ignores Internet structure. At its most elemental, Internet traffic on a wire consists of packets arriving through time, a marked point process. The point process is the packet arrival times, and the mark at each point is multivariate: all of the information in the TCP/IP headers. And since each packet traverses the Internet end-to-end, we must add the vast Internet topology to the structure. But just how much of the structure we want to invoke in a statistical model will depend on the goal and on the practicality.

One thing does however seem clear. If we want to study an aspect of the network that requires a model to account for TCP feedback at an individual connection level, then statistical models alone are unlikely to be able to account for packets properly. But linking up statistical models with network simulators could produce a highly effective hybrid (Joo, Ribeiro, Feldman, Gilbert, and Willinger 1999). Network simulators such NS (McCanne and Floyd 1998) are big a achievement of Internet analysis. To run the simulator, a network with routers and router algorithms is specified. TCP is simulated over this network and packets are produced. There are two places where statistics can help. One is statistical models of source traffic, which serve as input to the simulation. The other is statistical models for traffic aspects such as packet inter-arrival times and byte counts, built to reflect the characteristics of data on live Internet wires; the models can be used to validate the simulator output.

4. Very Large Data Bases

Packet header collection can result in a very large database, or VLDB, because packets come across Internet wires continuously. Even if the total throughput on a wire is small, the data grow and grow, so the database eventually gets large if collection continues. Our Bell Labs wire is just a trickle compared with high throughput wires of Internet ISPs. But after one year of collection, our databases of 328 million TCP connection flows with 6866 million TCP/IP packet-header pairs took up about 350 gigabytes. An Internet wire with 100 times the throughput of the Bell Labs wire would reach the same size in about 3.6 days.

S-Net: A Low-Cost, Distributed Data Analysis Computing Environment

The success of analyzing Internet traffic data depends heavily on an ability to analyze the traffic VLDB in great detail. We need to explore the raw data in its full complexity; relying only on summaries is inadequate. We need to study packet-level processes taking many variables into account; studying only byte counts in equally spaced intervals is inadequate. Success in detailed, intensive analysis depends on the analyst's computing environment.

To cope with the VLDB created by the Bell Labs traffic measurements, we developed S-Net (Cleveland, Lin, and Sun 2000b), a traffic collection and analysis system that begins with packet collection on a network link and ends with data analysis on a cluster of linux PCs running S, a language and system for organizing, visualizing, and analyzing data (Chambers 1998). Packet capture employs the program `tcpdump` (Jacobson, Leres, and McCanne 1998) running on a PC with Berkeley Unix, an altered kernel to enhance performance, time-stamping based on GPS clock discipline, and attention to packet drops. The compressed header files are moved to the cluster of linux PCs, which are linked by fast switches. Each PC has 1, 2 or 4 processors, and they all have large amounts of disk space. An algorithm then organizes the header information by TCP connection flow, and the flows are processed to create flow objects in S. Analysis is carried out in S.

Flows and S flow objects are computed in parallel on all of the PC processors and are stored on the disks of all machines. S is run on high-end PCs with large amounts of memory. Each analyst has a low-end PC that stores that user's S directories. The analyst logs onto a high-end machine from the home machine to run S, mounting the home S directories as well as the directories across the cluster housing the S objects. In other words, each data analysis session is distributed across the cluster.

S-Net has worked quite well. Because the PCs and switches can be inexpensive and linux is free, the cluster has a low overall cost. The cluster architecture scales readily; in our case, PCs and disks have been added and replaced incrementally as our database has grown. The S flow objects vary according to the specific analysis tasks; each is designed to enhance computational performance and to make the S commands that carry out the analysis as simple as possible. S is well suited to the task of analyzing Internet traffic data; its elegant design, which won it the ACM Software System Award for 1999, allows very rapid development of new tools.

But surely we can do better than S-Net. We need a whole new architecture for software for data analysis in networked environments that takes into account, from the ground up, the distributed nature of the environment. One effort is underway, the Omega Project (www.omegahat.org), and if it succeeds, data analysis of kinds, including Internet traffic modeling, will benefit.

Visualization Tools: MPMP Displays

As for most databases, visualization tools are vital for analyzing a VLDB. Analytic visualization tools support model development. A vital aspect is screen real estate. Because Internet databases are large and the structure is complex, we must accept the notion that displays need to cover tens and perhaps hundreds of pages with many panels on each page. Data visualization is often limited to a display of a set of data that can be placed all at once in our visual field. So it can be shocking at first to contemplate looking at so many pages. But using the structure of trellis display, it is easy to generate many pages (Becker, Cleveland, and Shyu 1996). And using a document viewer, it is possible to learn a great deal about Internet traffic data from from these *multipage, multipanel*, or *MPMP*, displays. Figure 1 is a trellis display with one page and two panels; it shows the HTTP start times for 2 of 10704 blocks. It was immensely informative for modeling the start times to see such displays for hundreds of blocks. And the only practical medium to communicate these MPMP displays is our medium of study, the Internet.

Synchronized Measurement

We can raise the measurement bar even higher. Suppose traffic is measured on two or more wires that have some traffic in common, and time is measured accurately, perhaps by using a feed from the satellite of a Global Positioning System. Such synchronized measurements can reveal much about the movement of traffic across the Internet (Paxson 1997b). But there has been very little synchronized collection because it greatly increases the already substantial burden of measurement, database management, and data analysis.

5. Conclusions

Packet header collection with timestamps, and TCP connection flow formation provide an effective framework for measuring traffic. But the resulting data are ferocious. They are non-stationary, long-range persistent, and distributions can have immensely long upper tails. There are many header variables such as the source and destination IP addresses, and there are many variables that can be derived from the header variables such as the throughput. The structure of these variables is complex. Added to this is the vastness of the Internet topology, and the intricacy and feedback of Internet protocols.

Challenges for producing tools and models for meeting the ferocity abound. Here is a short list: Statistical tools and models for point processes, marked point processes, and time series that account for nonstationarity, persistence, and distributions with long upper tails. • Frameworks for incorporating the structure of the Internet into traffic models and analyses. • Theoretical and empirical exploitations of the superposition of Internet traffic. • Methods for measuring and characterizing vast, complex network topologies. • Integration of statistical models with network simulators. • Synchronized network measurement, and tools and models for comprehending the results. • Methods for viewing MPMP data displays. • Low-cost, distributed computing environments for the analysis of very large databases.

Acknowledgements

Many thanks to Mark Hansen, Jim Landwehr, and Diane Lambert whose comments were very helpful for formulating and executing this writing.

References

Barford, P. and M. Crovella (1998). Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Proc. ACM Sigmetrics '98*, pp. 151–160.

Becker, R. A., W. S. Cleveland, and M. J. Shyu (1996). The Design and Control of Trellis Display. *Journal of Computational and Statistical Graphics 5*, 123–155.

Caceres, R., P. Danzig, S. Jamin, and D. Mitzel (1991). Characteristics of Wide-Area TCP/IP Conversations. In *Proc. of ACM SIGCOMM '91*, pp. 101–112.

Cao, J., W. S. Cleveland, and D. X. Sun (2000). On the Nonstationarity of Internet Traffic. Technical report, Bell Labs, Murray Hill, NJ.

Chambers, J. M. (1998). *Programming with Data*. New York: Springer.

Claffy, K., H.-W. Braun, and G. Polyzos (1995). A Parameterizable Methodology for Internet Traffic Flow Profiling. *IEEE Journal on Selected Areas in Communications 13*, 1481–1494.

Cleveland, W. S., D. Lin, and D. X. Sun (2000a). Network Simulation: Modeling the Nonstationarity and Long-Range Dependence of Client TCP Connection Start Times Under HTTP. In *Proc. ACM Sigmetrics '00*. To appear.

Cleveland, W. S., D. Lin, and D. X. Sun (2000b). S-Net: An Internet Traffic Collection and Analysis System. Technical report, Bell Labs, Murray Hill, NJ.

Crovella, M. E. and A. Bestavros (1996). Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In *Proc. ACM SIGMETRICS '96*, pp. 160–169.

Erramilli, A. O., O. Narayan, and W. Willinger (1996). Experimental Queueing Analysis with Long-Range Dependent Packet Traffic. *IEEE/ACM Transactions on Networking 4*, 209–223.

Feldman, A., A. A. Gilbert, and W. Willinger (1998). Data Networks as Cascades: Explaining the Mulifractal Nature of Internet WAN Traffic. In *Proc. ACM SIGCOMM '98*, pp. 42–55.

Feldmann, A. (1998). Characteristics of TCP Connection Arrivals . Technical report, AT&T Labs Research.

Floyd, S. and V. Paxson (1999). Why We Don't Know How to Simulate the Internet. Technical report, LBL Network Research Group.

Haslett, J. and A. Raftery (1989). Space-Time Modeling with Long Memory Dependence: Assessing Ireland's Wind Power Resource (with discussion). *Jour. of the Royal Statistical Society, C — Applied Statistics 38*, 1–50.

Hosking, J. R. M. (1981). Fractional Differencing. *Biometrika 68*, 165–176.

Jacobson, V., C. Leres, and S. McCanne (1998). tcpdump-3.4. http://ftp.ee.lbl.gov/nrg.html.

Joo, Y., V. Ribeiro, A. Feldman, A. C. Gilbert, and W. Willinger (1999). On the Impact of Variability on the Buffer Dynamics in IP Networks. In *Proc. 37th Annual Allerton Conference on Communication, Control, and Computing*. To appear.

Leland, W., M. Taqqu, W. Willinger, and D. Wilson (1994). On the Self-Similar Nature of Ethernet Traffic. *IEEE/ACM Transactions on Networking 2*, 1–15.

Mandelbrot, B. B. (1968). Noah, Joseph, and Operational Hydrology. *Water Sciences Research 4*, 909–918.

McCanne, S. and S. Floyd (1998). UCB/LBNL Network Simulator - ns (version 2). http://www-mash.cs.berkeley.edu/ns/.

Mogul, J. (1992). Observing TCP Dynamics in Real Networks. In *Proc. ACM SIGCOMM '92*, pp. 305–317.

Paxson, V. (1997a). Automated Packet Trace Analysis of TCP Implementations. In *Proc. ACM SIGCOMM*, pp. 167–179.

Paxson, V. (1997b). End-to-End Internet Packet Dynamics. In *Proc. ACM SIGCOMM '97*, pp. 139–152.

Paxson, V. and S. Floyd (1995). Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking 3*, 226–244.

Ribeiro, V. J., R. H. Riedi, M. S. Crouse, and R. G. Baraniuk (1999). Simulation of NonGaussian Long-Range-Dependent Traffic Using Wavelets. In *Proc. ACM SIGMETRICS '99*, pp. 1–12.

Riedi, R. and J. L. Vehel (1997). Multifractal Properties of TCP Traffic: A Numerical Study. Technical report, DSP Group, Rice University.

Ryu, B. K. and A. Elwalid (1996). The Importance of Long-Range Dependence of VBR Traffic in ATM Traffic Engineering: Myths and Realities. In *Proc. ACM SIGCOMM '96*, pp. 3–14.

Willinger, W. and V. Paxson (1997). Discussion of Heavy Tailed Modeling and Teletraffic Data by Sidney I.Resnick. *The Annals of Statistics 25*, 1805–1869.

Willinger, W., M. S. Taqqu, W. E. Leland, and D. V. Wilson (1995). Self-Similarity in High-Speed Packet Traffic: Analysis and Modeling of Ethernet Traffic Measurements. *Statistical Science 10*, 67–85.

Willinger, W., M. S. Taqqu, R. Sherman, and D. V. Wilson (1997). Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level. *IEEE/ACM Transactions on Networking 5*, 71–86.