Statistical Clustering of Internet Communication Patterns

Félix Hernández-Campos F. Donelson Smith
Kevin Jeffay
Department of Computer Science
University of North Carolina at Chapel Hill
{fhernand,smithfd,jeffay}@cs.unc.edu

Andrew B. Nobel
Department of Statistics
University of North Carolina at Chapel Hill
nobel@email.unc.edu

Abstract

We describe a new methodology for analyzing Internet traffic based on an abstract model of application-level communication and statistical cluster analysis, and argue that this method of analysis can serve as a foundation for building flexible traffic generation tools. We present the details of two case studies in which the new analysis tools are applied to data from the University of North Carolina at Chapel Hill and Internet2.

1 Introduction

The rigorous evaluation of new Internet technologies and changes in existing technologies requires careful simulation in controlled environments. A critical component of such simulations is the realistic reproduction of the workload of the Internet, *i.e.*, Internet traffic. However, modeling and synthetic reproduction of Internet traffic are themselves major challenges. Over the last two decades, the Internet has emerged as the most flexible communication infrastructure in the world. Hundreds of millions of hosts continuously exchange information using a broad spectrum of applications and protocols which exhibit a wide range of communication patterns. In this paper, we seek to address and study the heterogeneity of Internet traffic by combining an abstract model of application level communication with statistical cluster analysis. A central goal of our analysis is to enable new and more flexible approaches for traffic modeling and generation.

Internet traffic is composed of data exchanges between two end-points, driven by the details of a particular application, such as web browsing or electronic mail. Data exchanges are carried out by packets that travel through a world-wide fabric of network links and store-and-forward routers. Routers and links in the Internet see packet arrival processes in which packets from different data exchanges combine to form a heterogeneous *traffic mix*. For example, Table 1 summarizes the types of network traffic that were observed on Internet2 [19] during the week of November 4, 2002. The second row shows that World Wide Web data exchanges accounted for only 5.75% of the bytes in this network (13.76 Terabytes), and 5.89% of the

Application Type	Bytes		Packets	
Newsgroups	15.84%	37.90T	9.93%	36.76G
World Wide Web	5.75%	13.76T	5.89%	21.79G
File Transfer Protocol	3.02%	7.226T	2.70%	9.985G
Rsync	0.28%	665.6G	0.26%	950.9M
File Sharing	21.37%	51.15T	22.79%	84.39G
Audio/Video	3.56%	8.517T	2.97%	11.00G
Misc	2.35%	5.623T	4.66%	17.24G
Encrypted Traffic	2.22%	5.311T	1.65%	6.111G
Games	1.89%	4.512T	2.45%	9.071G
Advanced Apps	1.04%	2.486T	0.88%	3.253G
Measurement	0.84%	2.019T	1.03%	3.811G
Unidentified	41.85%	100.1T	44.80%	165.9G
Total	100.00%	239.3T	100.00%	370.2G

Table 1: *Traffic mix observed on Internet2 during the week of November 4, 2002 [5]. The first four rows correspond to individual applications, while the remaining ones report on groups of applications.*

individual packets (21.79 Gigapackets). Other rows provide information for groups of applications that have a similar purpose. For example, the fifth row includes traffic from file-sharing applications, such as Gnutella, eDonkey and FastTrack, which accounted for a large percentage of the total traffic. In summary, traffic mixes combine many different applications, that exchange data according to specific patterns.

The composition and type of traffic found on a link in the Internet vary with the location and size of that link, as well the time of day. For example, the traffic in the link that connects the University of North Carolina at Chapel Hill with the Internet is typically dominated by web browsing, which amounts for 50% of the total number of bytes transmitted, and by File Transfer Protocol data, which amounts for 25%. The traffic at other network locations can be substantially different. For example, Sprint's backbone links carry a much lower percentage of FTP data traffic, and a much higher percentage of traffic devoted to file-sharing applications, such as Kazaa [13]. By contrast, Table 1 shows that newsgroups traffic is much more prominent in Internet2 than in the other two networks. To be of broad utility, a method for analyzing or generating network traffic must account for this heterogeneity in a systematic fashion.

Concerning traffic generation, one approach is to separately model each individual application. However, given the large number of application currently in use, this is not practical. Here we take a different approach (see Section 2). As an alternative to application specific models, we define an abstract communication model that captures the pattern of data exchange in an abstract manner. This is sufficient to capture the workload of the Internet for a wide range of applications. In addition, the simplicity of the model makes it feasible to develop an efficient techniques to analyze a large number of data sets from many different network locations.

Our abstract description of Internet communication patterns enables the study of the population of connections observed on the Internet and the development of a taxonomy of the most important patterns. We propose statistical clustering as the basic technique for this study, and Section 3 describes our methodology. Section 4

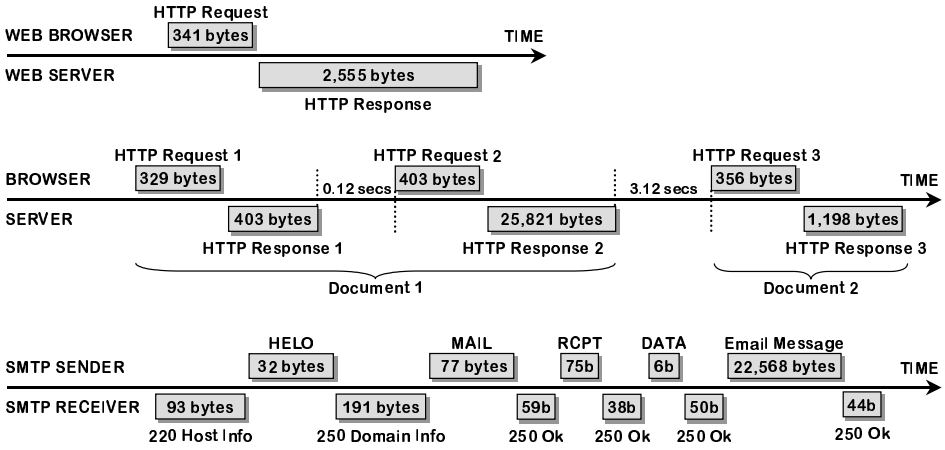

Figure 1: *Examples of application-level data exchanges. From top to bottom, non-persistent HTTP connection, persistent HTTP connection and SMTP connection.*

presents two examples of traffic clustering. Finally, Section 5 reviews related work.

2 Abstract Communication Model

Our approach to the synthetic generation of network traffic is based on treating traffic sources as network-independent entities. We begin with the observation that, from the perspective of the network, the vast majority of application-level protocols are based on a small number of data exchange patterns that occur within a logical connection between the endpoint processes. In our model, the two endpoint processes exchange data in units defined by their specific application-level protocol. In our model, two endpoint processes exchange data in a sequential fashion as they execute a particular application. The transferred data is summarized by *application-data units* (ADU's) whose size depends on the application protocol and the data objects used in the application. For example, a common type of interaction between a web browser and a web server using the HyperText Transfer Protocol (HTTP) consists of a single exchange of data, as shown in the upper diagram in Figure 1. The browser first opens a connection to the server and sends a request for an specific object (*e.g.*, an HTML page or an image). This request is the first ADU in the data exchange. After the server receives the entire request, it replies with the requested object and closes the connection. This object (the response) is the second ADU in this connection. The sizes of the request and the response do not depend on network characteristics, such as the time required to send a packet from the browser to the server or the bandwidth available between these two hosts. As a consequence, we simply model the data exchange in this example as a pair of data unit sizes (341 bytes, 2555 bytes). In a simulation, this communication pattern can be reproduced by establishing a connection between two end-points and then exchanging the data units in the specified order[1].

In the general form of our abstract communication model, which we call the

[1]Note that this model assumes causality in the exchange of data, in the sense that the first data unit, the *request*, must be received in its entirety before the second data unit, the *response*, is sent in the opposite direction.

a-b-t model, each point-to-point communication can be represented as a *connection vector* $(c_1, \ldots, c_n)$. Here n is the total number of *epochs* in the connection and $c_j = (a_j, b_j, t_j)$ is a triplet describing the data (a_j, b_j) and time (t_j) parameters of the j'th transmission epoch. Each a_i captures the amount of data sent from the initiator of the communication (*e.g.*, a web browser) to the other end-point, while each b_i represents data flowing in the reverse direction. In the previous example, the data exchange between the web browser and the web server would be summarized as a vector with a single epoch ($n = 1$), in which a_1 equals 341 and b_1 equals 2,555. The time parameters t_j are used to model quiet times between data exchanges, that, if sufficiently long, represent application-level behavior, such as human *think times* and long processing delays.

Figure 1 shows, in a compact graphical representation, three examples of connections that were captured on the main Internet link at the University of North Carolina at Chapel Hill. The first one, as mentioned above, corresponds to a single data exchange between a web server and a web browser. The second example illustrates a persistent HTTP connection, in which browser and server exchange three pairs of data units. The representation of this connection in the a-b-t model is

$$((329, 403, 0), (403, 25821, 3.12), (356, 1198, \phi))$$

We generally ignore values of t_j below an *idle time threshold* $\tau = 1$ second, since we are only interested in capturing inter-epoch times that are clearly network-independent. Notice that t_3 is not an inter-epoch time, so we mark it as ϕ. The third example shows a Simple Mail Transfer Protocol (SMTP) connection established between two email servers, that may be represented in the a-b-t model as

$$((0, 93, 0), (32, 191, 0), (77, 59, 0), (75, 38, 0), (6, 50, 0), (22568, 44, \phi))$$

In this protocol, the two servers engage in a conversation that includes a number of small data units, such as b_4, which carries the email address of the intended recipient, and a single large large data unit, a_6, with the text of the email. Notice that the first data unit in this example is not sent by the connection initiator, making a_1 equal to 0.

The patterns of data exchange shown in Figure 1 correspond to applications that use the Transport Control Protocol (TCP) as their underlying process-to-process communication protocol. This protocol, used in the vast majority of the data exchanges on the Internet, carries both application data and control data. Control data includes sequence numbers and other bookkeeping information in each packet. Some packets, such as those used to establish a TCP connection, do not carry any application data. Application behavior is independent of control data, which is transparently handled by the operating system. Consequently, the a-b-t model does not capture control data, since we are only concerned with modeling the way applications exchange data. As a consequence, we can use the same workload, modeled as a set of connection vectors, for comparing the performance of two network mechanisms. For example, we can compare two *flavors* of TCP that employ different control strategies.

A further refinement of the model is that time intervals longer than τ in which there is no activity within a connection also define exchange boundaries. This helps capture unidirectional communications patterns. For example, some applications refresh the state of the clients periodically. Without an idle time threshold, this unidirectional sequence of updates would be combined into a single epoch, with a large value of a_i or b_i, rather than a sequence of epochs. In addition, the inactivity threshold overcomes a potential limitation of the model, which does not

capture times between successive a- and b-type data units; if the time between two such data units is larger than τ, then they are placed in separate epochs, resulting in a subconnection of the form $((a_i, 0, t_i), (0, b_{i+1}, t_{i+1}))$. This properly reflects a prolonged time between a request and its response, and similar application level characteristics of other communication patterns.

The simplicity of the a-b-t model makes it possible to convert any connection[2] into a connection vector by looking only at transport-level headers of packets. We have developed a tool to convert any TCP/IP protocol header trace into a set of connection vectors. This tool, based on a generalization of the techniques presented in [33], requires only one direction of the packet trace (*i.e.*, segments flowing from the connection initiator to the other end-point, or vice versa), making large-scale data acquisition possible. We have successfully applied our techniques to publicly available traces, such as those provided by NLANR [26]. In addition, we are using this approach to analyze our own collection of packet header traces obtained at the link that connects the University of North Carolina at Chapel Hill and the Internet. This data set is one of the largest currently available (more than 2 Terabytes), and it includes traces collected during the last 4 years.

3 Clustering Communication Patterns

The a-b-t model provides a framework for the systematic identification and study of application-level communication patterns in Internet traffic. Traffic modeling, sampling techniques, and other research areas within networking can certainly benefit from the analysis and classification of such patterns. For example, the performance of transport protocols depends heavily on the patterns of data exchange within transport connections, so a good understanding of these patterns and their impact is needed for balancing among the tradeoffs that exist in the design of these protocols. For instance, TCP can be tuned to provide better performance for transferring small data units at the price of higher instability and less fair allocation of bandwidth. We can analyze the real benefits of this tuning using simulations, helping to decide whether this change of TCP's parameters is benefitial or not for the Internet as a whole. Only those simulations that make use of a broad and representative set of data exchange patterns in their workloads can help to draw general conclusions the effectiveness of new network mechanisms.

Statistical clustering techniques, see *e.g.*, [14, 20, 9], provide a useful and flexible tool for grouping connections into traffic classes that represent similar communication patterns. Formally, a clustering scheme is a procedure that divides a given set of feature vectors $x_1, x_2, \ldots, x_m \in \mathbb{R}$ into k disjoint groups $S_1, S_2, \ldots, S_k$, which are known as clusters. The goal of clustering is to find a small number k of clusters such that feature vectors within the same clusters are close together, while vectors in different clusters are far apart. In our approach, each application-level connection vector derived from a transport connection is first summarized using a vector of *statistical features*. Each feature captures some relevant characteristic of the sequence, such as the number of exchanges, the total number of bytes send by the initiator, the homogeneity in the sizes of the data units, and so on. We then measure the similarity between two connection vectors by the similarity between their associated feature vectors. We consider two alternative distance measures (see

[2]To be more precise, concurrent data exchanges are not supported in this model. We are considering extensions of the model to accomodate this type of communication, which is generally much less frequent than non-concurrent data exchange.

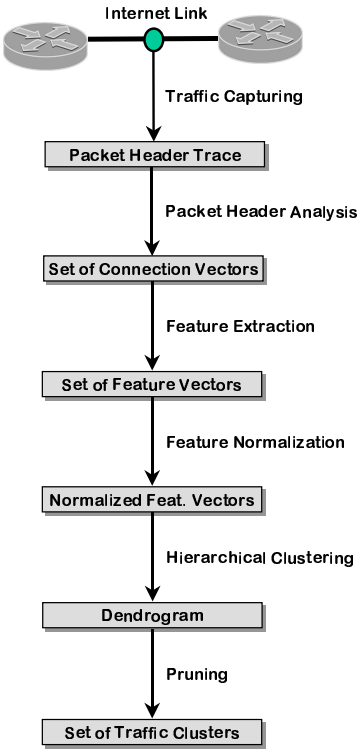

Figure 2: *Overview of our approach for clustering Internet communication patterns.*

[9]), the standard Euclidean distance, *i.e.*,

$$d_{ij} = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2},$$

and Pearson correlation coefficient. Once the distance between each pair of connection vectors has been defined, these vectors can be grouped using any number of standard clustering algorithms. We have applied a number of different clustering schemes to our data, but have focused on agglomerative and divisive hierarchical methods. These methods have proven to be effective, and their graphical representation as trees (dendrograms) provides a useful way of identifying and analyzing groups of related communication patterns. Figure 2 provides an overview of the basic steps in our methodology, which are described in greater detail in the rest of this section.

As a first step in clustering source level communication patterns, we extract from each connection vector a number of numerical features that are designed to capture important aspects of the two-way data transfer it describes. Let $v = (c_1, \ldots, c_n)$ be a given connection vector whose j'th epoch is given by the triple $c_j = (a_j, b_j, t_j)$, as described above. The most critical features of v are the number of epochs, denoted by e, and the total number of bytes sent by each of the connection hosts, $a_{tot} = \sum_{j=1}^{n} a_j$ and $b_{tot} = \sum_{j=1}^{n} b_j$. Let $A = \{a_1, \ldots, a_n\}$ be the collection of a-type data units measured during the connection. Other useful features include

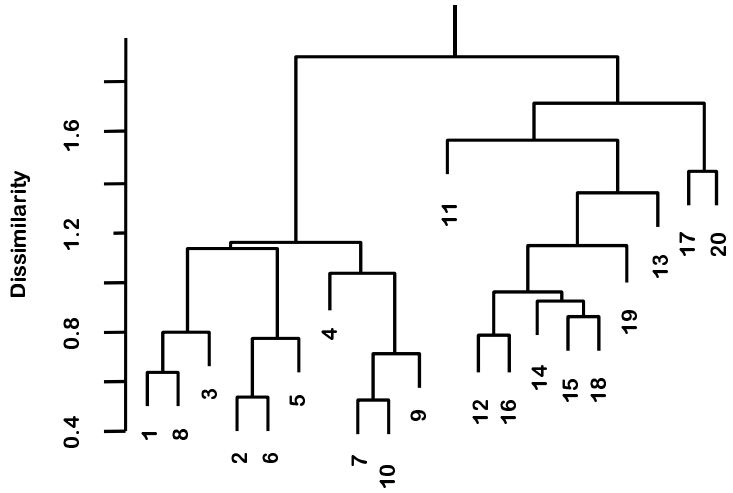

Figure 3: *Result of clustering a training set of 20 connections using agglomerative hierarchical clustering. Leaves labeled from 1 to 10 correspond to Telnet connections, while those labeled from 11 to 20 correspond to HTTP connections.*

$a_{max} = \max\{a_j \in A\}$ and $a_{min} = \min\{a_j \in A\}$, the mean a_μ and standard deviation a_σ of A; and the first, second and third quartiles of A, denoted by a_{1q}, a_{2q} and a_{3q} respectively. In order to better capture the sequential structure of the a-type data units, we measure the total variation $a_{vs} = \sum_{j=2}^{n} |a_j - a_{j-1}|$, maximum first difference $a_{fd} = \max_j |a_j - a_{j-1}|$, lag-1 autocorrelation a_ρ, and homogeneity $a_h = (a_{max} + 1)/(a_{min} + 1)$ of $a_1, \ldots, a_n$ in cases where $n \geq 2$. Analogous features can be extracted from the collection $B = \{b_1, \ldots, b_n\}$ of b-type data units. As inter-epoch times will reflect network, as well as application-level behavior, we restrict our attention to t_{max}, t_{2q}, and t_{tot}.

To assess the relationship between the a- and b-type data units, we also measure directionality $dir = \log(a_{tot}/b_{tot})$ and the lag 0 and 1 cross-correlations between B and A, denoted ρ_1 and ρ_2 respectively. In our preliminary analysis we found that rank correlations exhibited a a more diverse and meaningful spectrum of values across different connections. Thus all correlation measurements are based on Spearman's rank correlation coefficient, rather than the more common Pearson coefficient.

The features defined above provide us with a reasonable starting point for our cluster analysis of connections, but they are not the final word. The selection of new features, and the refinement (or possibly elimination) of existing ones, is a subject of current research. Different sets of features are used in the examples given below.

Whichever features one ultimately chooses, there are a number of practical issues that need to be addressed before they can profitably be used to cluster connections. The first issue involves scale. While correlations will range between -1 and $+1$, features such as e and a_{tot} can range anywhere from one to several million. To address this disparity, we first take logarithms of those features that vary over several orders of magnitude. Each feature is then translated and scaled so that, for the vast majority (more than 96%) of measured connections, its value is between 0 and 1. In exceptional cases, *e.g.*, a connection with 10^7 epochs, we allow features

Feature			Description
n			Number of epochs
a_{tot}	b_{tot}		Total bytes
a_{max}	b_{max}	t_{max}	Maximum bytes or seconds
a_{min}	b_{min}		Minimum bytes
a_{μ}	b_{μ}		Mean bytes
a_{σ}	b_{σ}		Standard deviation
a_{1q}	b_{1q}		First quartile
a_{2q}	b_{2q}		Second quartile
a_{3q}	b_{3q}		Third quartile
a_{vs}	b_{vs}		Total variation
a_h	b_h		Homogeneity $(a_{max}+1)/(a_{min}+1))$
a_{ρ}	b_{ρ}		Lag-1 autocorrelation
$\rho_1(a_{1..n}, b_{1..n})$			Spearman's Rank Correlation
$\rho_2(b_{1..n-1}, a_{2..n})$			Spearman's Rank Correlation with Lag 1

Table 2: *The 26 statistical features used in the divisive hierarchical clustering example shown in Figure 4.*

greater than 1 or less than 0. Allowing features to take values outside the unit interval avoids the possible compression of their dynamic range by a small fraction of outliers.

Once normalized, each feature plays a role in determining the Euclidean distance between two feature vectors. One may weight the contributions of different features differently, but we have not done this in our experiments. A second practical issue is that some features (*e.g.*, correlations and total variation) are not well-defined or not meaningful for connection vectors with fewer than three epochs. When comparing a connection with ten epochs to one with two epochs, we look only at the Euclidean distance (or correlation) between those features that are defined in both associated vectors, and then normalize by the number of such "active" features, so that the resulting distance can be compared to distances between longer connections.

We initially tested our approach by clustering training data sets with a small number of connections. Figure 3 shows the result of clustering 20 connections collected at the University of North Carolina at Chapel Hill. We analyzed this data set using divisive hierarchical clustering as implemented in R [17], after converting each connection vector into a feature vector that included all of the statistical features described above. Ten of the connections in the data set carried Telnet traffic (*i.e.*, interactive remote shell), while the other ten carried web traffic. The communication patterns used by these two protocols are quite different, so appropriate clustering should be able to split the data set into two subpopulations. As shown in the figure, two distinct clusters, emanating from the root of the dendrogram, are readily apparent.

4 Clustering Examples

4.1 Divisive Hierarchical Clustering Example

In our first example of clustering traffic, we study a packet header traces collected during April 2002 at the main network link that connects the University of North

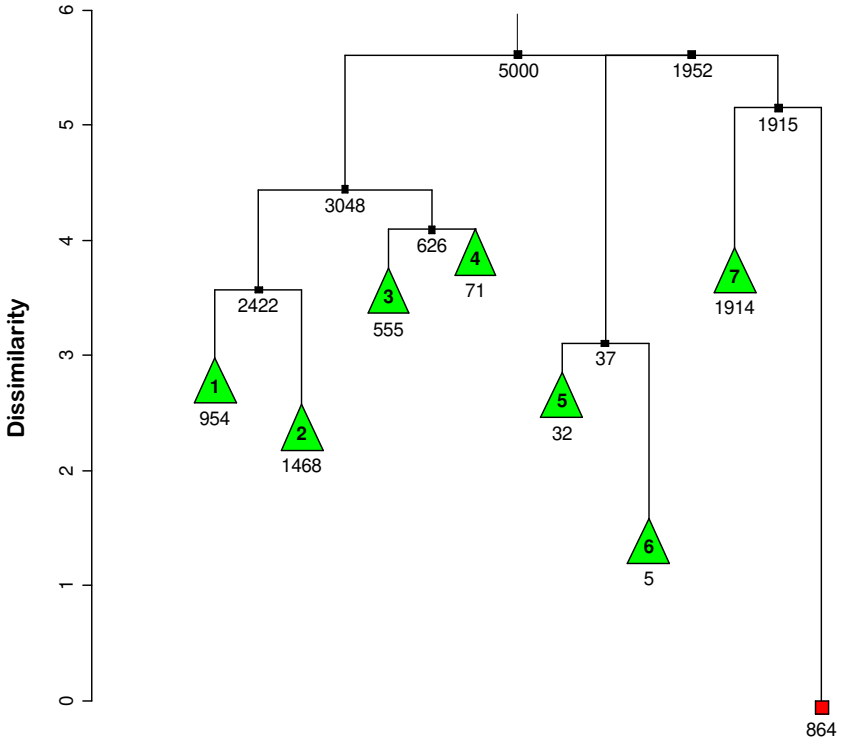

Figure 4: *Dendrogram obtained from the divisive hierarchical clustering of data set of 5,000 connections, pruned at depth 4.*

Carolina at Chapel Hill and the Internet. We first converted this trace into a set of several million connection vectors, from which we drew a random sample of 5,000 connection[3] vectors with 2 epochs or more. We then computed the feature vectors of the connections in this sample, using the features reported in Table 2. After normalizing the feature vectors, we analyzed them using the `diana` procedure with Euclidean distance as implemented R [17], which follows the algorithm described in [22].

The result of clustering the set of 5,000 are shown in Figure 4, using a new visualization function that we implemented in the R language. The dendrogram shown is the result of pruning the full dendrogram at depth 4. The plot depicts pruned internal nodes as green triangles with a cluster number, and leaves as red squares with a connection vector number below them. Each internal node is annotated with the number of connection vectors grouped under its branches. For example, the root of the tree is annotated with 5,000, since all of the connection vectors fall under this internal node. The first triangle on the left, marked as cluster number 1, groups 954 connection vectors. The height of each internal node corresponds to the dissimilarity between its children.

The dendrogram reveals some useful structure in the set of connection. Connections in cluster 1 mostly correspond to HTTP, HTTPS (encrypted web traffic) and AOL traffic, while those in cluster 3 correspond to mail transfer protocols, such

[3]This number is relatively small due to computational difficulties, but it should be sufficient to identify the most important clusters in the full data set.

Feature			Description
n			Number of epochs
a_{tot}	b_{tot}	t_{tot}	Total bytes or seconds
a_{2q}	b_{2q}	t_{2q}	Second quartile (Median)
a_{fd}	b_{fd}		Maximum first difference
a_h	b_h		Homogeneity $(a_{max}+1)/(a_{min}+1))$
dir			Directionality $(\log(a_{tot}/b_{tot}))$
$\rho_1(a_{1..n}, b_{1..n})$			Spearman's Rank Correlation
$\rho_2(b_{1..n-1}, a_{2..n})$			Spearman's Rank Correlation with Lag 1

Table 3: *The 14 statistical features used in the agglomerative hierarchical clustering example illustrated in Figure 5.*

as SMTP and the Post Office Protocol (POP). The clustering algorithm accurately separated two clearly different communication patterns. Clusters 5 and 6 include connections in which all the b-type data units are zero, and whose port numbers did not map to known applications. Finally, cluster 7 grouped together HTTP, HTTPS, Microsoft Directory Service and RTSP connections. The only leaf shown in the dendrogram (connection vector 864) was an FTP-DATA connection with $n = 2$, $a_{tot} = 50K$ and $b_{tot} = 0$.

While the reveled structure is suggestive, it is difficult to explain the observed hierarchy, and this motivated us to use a different tool in our more recent work (see the next subsection). Furthermore, computation of the full dendrogram was slow; this 5,000-connection example required many hours of processing time. Another difficulty we experienced is the $O(n^2)$ memory requirement, present in most statistical clustering algorithms, which comes from the need to compute the distance between each pair of connection vectors as the first step.

4.2 Agglomerative Hierarchical Clustering Example

We applied our methodology to the clustering of a sample of connections from the Abilene-I data set [24]. The sample consisted of 717 TCP connections[4]. Each connection was first transformed into a connection vector, and then summarized into a feature vector. The result was a matrix of 717 rows and 14 columns. Table 3 describes the 14 statistical features that were part of each vector.

Feature vectors were clustered using the average-linkage agglomerative method proposed by Sokal and Michener [34], with Pearson correlation coefficient as the similarity measure[5]. For this clustering, we employed the implementation of the algorithm and the visualization tool developed by Eisen *et al.* in the context of gene expression arrays (microarrays) [8]. The result of the clustering is shown in Figure 5. The colored array in the center of the figure is a heat map that represents the matrix of feature vectors. Each row in the array corresponds to one connection, and each column corresponds to one statistical feature. Therefore, the fourteen colored cells within a row represent the values of the statistical features of a single connection. Values are displayed using a scale of increasingly lighter shades of blue

[4]While this number is relatively small, we believe it is representative of the coarse-grained structure in the data set, and it makes it possible to include graphical output in this paper. We have applied our method to larger sets with up to 25,000 connections.

[5]We recently obtained similar results with a newer version of the software that support Euclidean distance

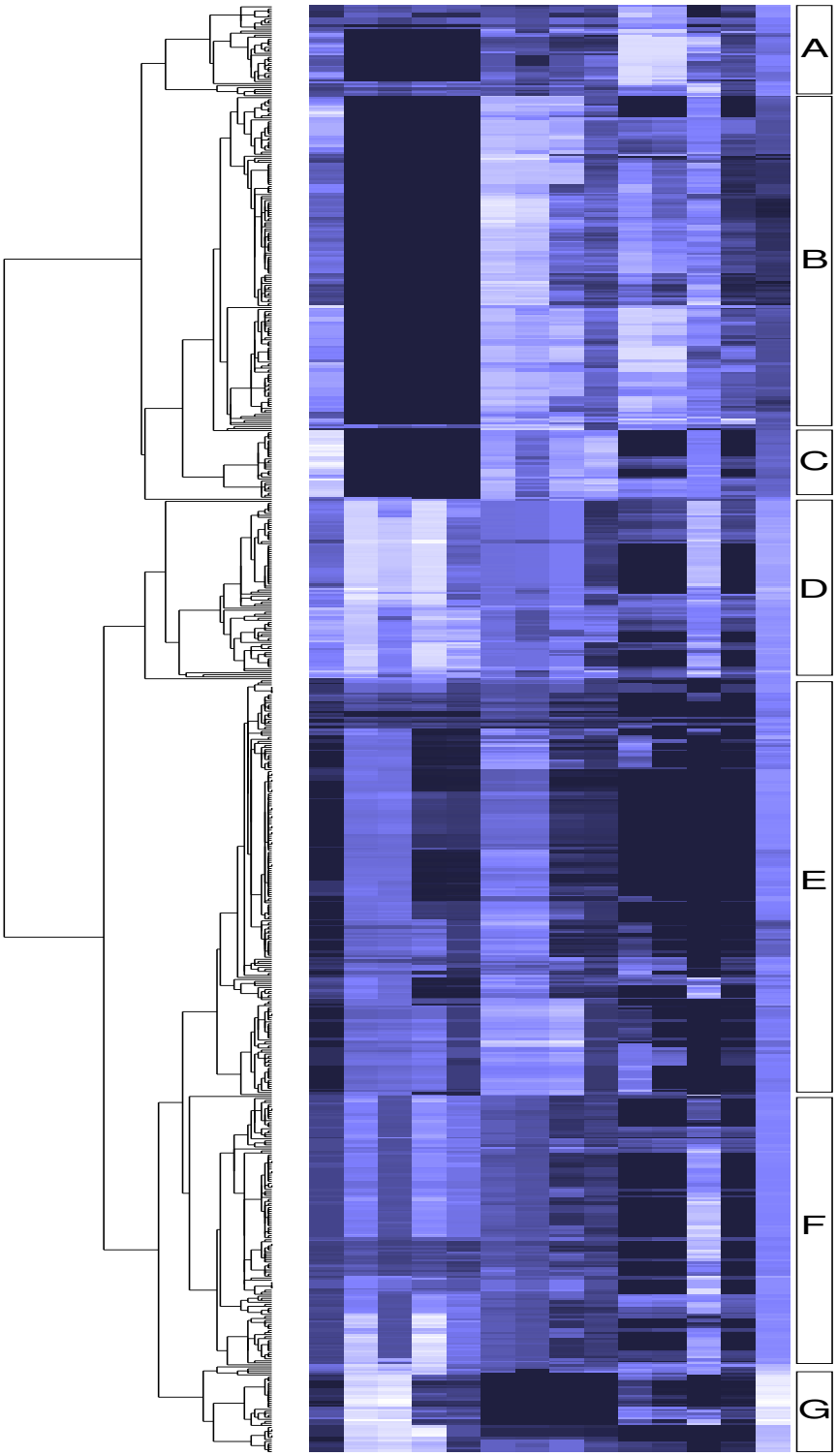

Figure 5: *Result of clustering a sample of connections from the Abilene-I data set. From left to right, the columns of the heat map correspond to* $n, a_{tot}, a_{2q}, a_{fd}, a_h, b_{tot}, b_{2q}, b_{fd}, b_h, t_{tot}, t_{2q}, dir, \rho_1$ and ρ_2.

(in other words, the larger the value, the lighter the color). On the left side of the array, a binary tree is used to display the hierarchical clustering of connections. Each internal node represents a group of connections, and these connections are divided among its two children. Each leaf represents an individual connection (they are hard to see at this scale). The height of an internal node represents the average dissimilarity between its children. On the right side of the array, seven rectangles (labeled from A to G) are used to highlight seven clusters that exhibit a high degree of internal cohesion (correlation is 0.6 or more) and substantial separation from other clusters (dissimilarity sharply increases when any of these clusters is joined to another cluster).

The interpretation of the resulting clusters confirms the effectiveness of our approach for grouping connections into homogeneous communication patterns. Clusters A and B group together connections with small a-type data units. By looking at the destination port numbers of these connections, we found that most correspond to file sharing applications, mainly Kazaa (port number 1214), eDonkey (4662), and Gnutella (6346). Connections in cluster A show substantially smaller b-type data units than those in cluster B, and they also exhibit much longer inter-exchange times. We believe that connections in the former cluster mainly correspond to file-sharing sessions in which only searches and no file downloads took place, while file downloads did occur in the connections grouped in the latter cluster. Some number of connections in these two clusters used other destination ports, such as 80, but their intra-connection dynamics did match those of file-sharing applications. These connections provide a good example of port number *hijacking*, a technique frequently employed to overcome firewalls and bandwidth caps.

Cluster C includes connections that have small a-type data units, and a number of exchanges that is significantly larger than that in the connections contained in clusters A and B. The destination port numbers correspond to a variety of applications, including Gnutella, HTTPS and Telnet.

Connections in cluster D are almost exclusively destined to port 119 (NNTP), and they show a clearly different pattern of data exchanges (large a-type data units and moderate b-type data units). Cluster E groups together connections destined to ports 80 (HTTP), 443 (HTTPS) and other ports that are also used for the web traffic, such as 8080 and 8443. Cluster F is mostly composed of SMTP connections (port 25) and some number of POP (110) and Oracle (1521). Finally, cluster G contains FTP-Data connections. Some of these connections used source port 20, but the vast majority used other dynamically-negotiated port numbers. We have confirmed that these connections carried FTP-Data traffic by verifying that parallel FTP-Control connections existed.

The seven clusters described above can be further explored and decomposed into subclusters, an operation naturally supported by the hierarchical structure of the binary tree. For instance, we found other smaller clusters that group together other types of communication dynamics, such as those exhibited by streaming media and FTP-Control connections.

5 Related Work

A compelling case for identifying traffic generation as one of the key challenges in Internet modeling and simulation is made by Floyd and Paxson in [11]. In particular, they emphasize the importance of generating traffic from source-level models. Two important measurement efforts that focused on application-specific traffic models, but which preceded the growth of the web, were conducted by Danzig *et al.* [7, 2, 7]

and by Paxson [31]. These researchers laid the foundation for TCP traffic modeling at the source level using empirical data to derive distributions for the key random variables that characterize traffic sources. A more recent study of Real Audio traffic by Mena and Heidemann [25] provides a source-level view of streaming-media, much of it carried on UDP flows.

Measurements to characterize web usage have become a very active area for research. Web traffic generators in use today are usually based on data from the two pioneering measurement projects that focused on capturing web-browsing behaviors: the Mah [23], and Crovella *et al.* [30, 6, 1] studies. Traffic generators based on both of these sources have been built into the widely used ns-2 network simulator [27], which has been used in a number of studies related to web-like traffic, *e.g.*, [28, 10]. These models have also been used to generate web-like traffic in laboratory networks [1]. More recent models of web traffic suitable for source-level traffic generation have been published by Smith *et al.* [33] and Cleveland *et al.* [4, 3]. Finally, we note that source modeling for multimedia data, especially variable bit-rate video, was an active area of research recently (see [12, 15, 21]). Other approaches to traffic source modeling with an emphasis on packet-level traffic are surveyed in [16].

A current research project with similar goals to ours is the SAMAN project [32] at ISI. Their goal is to "develop tools and approaches that integrate multi-point network measurements to rapidly generate compact, accurate, application-level models that are accurate across a wide range of time-scales." They have produced a software tool, RAMP, that "takes a live tcpdump trace from network and generates a set of CDF (Cumulative Distribution Function) files that model web and FTP traffic." Portions of their RAMP software is based on the trace analysis tools we developed at UNC for web traffic [33] (our tools are also distributed as part of the ns-2). Commercial synthetic traffic generation products such as Chariot [18] exist, however, these generators are typically based on a limited number of traffic source types. Moreover, it is not clear that any are based on empirical measurements of Internet traffic.

While not directly applicable as a traffic-generation method, modeling of packet-level arrival processes is an important element of our approach to validating synthetic traffic. The breakthrough results in this field identified self-similarity and long-range dependency as fundamental features of network traffic that must be reproduced in synthetic traffic. The book by Park and Willinger [29] (and the references therein) is an excellent resource and guide to the results in this area.

6 Conclusion

Our methodology for the study of data exchange patterns in transport connections provides an effective way of visualizing and clustering the behavior of Internet sources. We believe that the deeper understanding of Internet traffic that can be gained with our approach will help develop better traffic measurement and modeling techniques. We also believe that traffic workloads for testing and simulation should reflect the clearly distinguished patterns of communication uncovered by clustering data set of Internet connections.

Acknowledgments

We would like to thank the NLANR Measurement and Network Analysis Group for making their data and tools publicly available. Support for their work was

provided by the National Science Foundation (NSF) (cooperative agreement no. ANI-9807479). We are also indebted to Michael Eisen, for making his clustering and visualization tools freely available for academic researchers, and to the developers of the R language. We also thank Michele C. Weigle for helpful comments.

This work was supported in parts by the National Science Foundation (grants ITR-0082870, CCR-0208924, EIA-0303590, and ANI-0323648), Cisco Systems Inc., the IBM Corporation, and a doctoral fellowship from the Computer Measurement Group. Andrew Nobel was supported in part by NSF grant DMS 997 1964.

References

[1] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *Proc. of the ACM SIGMETRICS*, pages 151–160, 1998.

[2] Ramon Caceres, Peter B. Danzig, Sugih Jamin, and Danny J. Mitzel. Characteristics of wide-area TCP/IP conversations. In *Proc. of the conference on Communications architecture and protocols*, pages 101–112. ACM Press, 1991.

[3] Jin Cao, William S. Cleveland, Dong Lin, and Don X. Sun. On the nonstationarity of Internet traffic. In *Proc. of SIGMETRICS/Performance*, pages 102–112, 2001.

[4] William S. Cleveland, Dong Lin, and Don X. Sun. IP packet generation: statistical models for TCP start times based on connection-rate superposition. In *Measurement and Modeling of Computer Systems*, pages 166–177, 2000.

[5] Internet 2 Consortium. Internet2 Netflow Weekly Report: Week of 20021104, `http://netflow.internet2.edu/weekly/20021104`, November 2002.

[6] Mark E. Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[7] Peter B. Danzig and Sugih Jamin. tcplib: A library of TCP/IP traffic characteristics. *USC Networking and Distributed Systems Laboratory TR CS-SYS-91-01*, October, 1991.

[8] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci.*, 95:14863–14868, December 1998.

[9] Brian S. Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*. Arnold, 4th edition, 2001.

[10] W. Feng, D. Kandlur, D. Saha, and Kang G. Shin. BLUE: A new class of active queue management algorithms. Technical Report CSE-TR-387-99, University of Michigan, 15, 1999.

[11] S. Floyd and V. Paxson. Difficulties in simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, August 2001.

[12] Mark W. Garrett and Walter Willinger. Analysis, modeling and generation of self-similar VBR video traffic. In *SIGCOMM*, pages 269–280, 1994.

[13] Sprint Advanced Technology Laboratory (IP Group). IP monitoring project: Data management system, 2002. http://ipmon.sprintlabs.com.

[14] John A. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[15] D. Heyman and T.V. Lakshman. Source models for VBR broadcast video traffic. *IEEE/ACM Trans-actions on Networking*, 4(1):37–46, February 1996.

[16] Helmut Hlavacs, Gabriele Kotsis, and Christine Steinkellner. Traffic source modeling. Technical Report TR-99101, Institute of Applied Computer Science and Information Systems, University of Vienna, 1999.

[17] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[18] NetIQ Software Inc. Chariot performance evaluation platform. http://www.netiq.com/products/chr/default.asp.

[19] Internet2. http://www.internet2.edu.

[20] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990.

[21] E. W. Knightly and H. Zhang. D-bibd: An accurate traffic model for providing QoS guarantees to VBR traffic. *IEEE/ACM Transactions on Networking*, 5(2):219–231, April 1997.

[22] P. Macnaughton-Smith, W. T. Williams, M. B. Dale, and L. G. Mockett. Dissimilarity analysis: A new technique of hierarchical sub-division. *Nature*, 202:1034–1035, 1965.

[23] Bruce A. Mah. An empirical model of HTTP network traffic. In *Proc. of IEEE INFOCOM*, pages 592–600, 1997.

[24] NLANR Measurement and Network Analysis Group. Trace IPLS-CLEV-20020814-090000-0 (Abilene-I data set). http://pma.nlanr.net/Traces/long/ipls1.html.

[25] A. Mena and J. Heidemann. An empirical study of real audio traffic. In *Proceedings of IEEE INFOCOM '00*, March 2000.

[26] National laboratory for applied network research (NLANR). http://www.nlanr.net.

[27] ns-2 (Network Simulator). http://www.isi.edu/nsnam/ns/.

[28] Teunis J. Ott, T. V. Lakshman, and Larry H. Wong. SRED: Stabilized RED. In *Proceedings of INFOCOM*, volume 3, pages 1346–1355, 1999.

[29] K. Park and W. Willinger. *Self-Similar Network Traffic and Performance Evaluation*. Wiley, 2000.

[30] Kihong Park, Gitae Kim, and Mark Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. Technical Report 1996-016, Boston University, 30, 1996.

[31] Vern Paxson. Empirically derived analytic models of wide-area TCP connections. *IEEE/ACM Transactions on Networking*, 2(4):316–336, 1994.

[32] Simulation augmented by measurement and analysis for networks (SAMAN) project. `http://www.isi.edu/saman/`.

[33] F. Donelson Smith, Félix Hernández-Campos, Kevin Jeffay, and David Ott. What TCP/IP protocol headers can tell us about the web. In *Proc. of SIGMETRICS/Performance*, pages 245–256, 2001.

[34] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *Univ. Kansas Sci. Bull.*, 38:1409–1438, 1958.