Understanding Patterns of TCP Connection Usage with Statistical Clustering

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

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

Abstract

We describe a new methodology for understanding how applications use TCP to exchange data. Our approach is based on an abstract model of application-level communication that is suitable for statistical cluster analysis. We describe how to transform TCP connections into vectors of statistical features and how to cluster these vector using existing hierarchical clustering methods. We also adapt a visualization technique developed in the context of gene expression arrays to the analysis of network traffic, and demonstrate that this approach makes it possible to understand the results of the clustering in a way that is meaningful for networking research. Our methodology also provides the foundation for more flexible synthetic traffic generators and could be adapted to the analysis of workloads in other contexts.

1 Introduction

The Transport Control Protocol (TCP) provides the most common foundation for Internet applications. File-sharing, the web, email, instant messaging, and many other applications make use of the reliable transport service offered by TCP to communicate their data across the Internet. This wealth of applications represents an important challenge for the researchers, who generally need to have a good understanding of the structure of network traffic. Furthermore, it makes it very challenging to construct source-level traffic models that are suitable for theoretical analysis and for generating synthetic traffic in simulations and testbed environments.

Current TCP traffic is driven by a large, heterogenous collection of applications, and its is not generally dominated by web traffic. For example, traffic in Sprint's backbone as reported in [12] shows that only 20-40% of the traffic is due to the Web. Similarly, the weekly netflow report from Internet2 [4] reports an even smaller percentage of web traffic. Both backbones carry subtantial traffic driven by an ever-changing set of file-sharing applications, such as BitTorrent and Shoutcast, by gaming networks, such as Battlenet, and by traditional file transfers, emails, newgroups, etc. Internet applications make use of TCP in different ways, which range from opening a connection and sending a single file, to complicated exchanges of control messages and data objects.

The exact number of applications that contribute to these *traffic mixes* is not known, and even the actual impact of well-known protocols is unclear. This is due to the shortcomings of the state-of-the-art in traffic monitoring, which makes use of registered and well-known port numbers to classify packets and computer statistics. The problem is that new applications often do not use a registered port, do not have a fixed port number, or simply disguise themselves using the port number of another application (for example, the web's port 80) to avoid detection (so they can pass through firewalls, and avoid rate limits). As an example of the shortcomings of port numbers, both the Sprint and the Abilene reports leave a large percentage of the traffic unidentified (20-35%).

The behavior of some applications, such the web and some file-sharing application, is well-understood and has been the focus of significant modeling effort. Other applications that are newer or less common have not received much attention. Furthermore, applications evolve quickly, often making existing models obsolete. For example, the extension of HyperText Transfer Protocol (HTTP) to support persistent connections had a major impact in web traffic and its use of TCP. In addition, constructing models of application traffic is a difficult and time-consuming task, so most models are never updated after publication. For example, popular Simple Mail Transfer Protocol (STMP) models (*e.g.*, [6]) are quite old and predate the wide-spread use of broadband that has made large attachments far more common. The goal of our work is to develop more powerful and general techniques for understanding and modeling traffic that will enable us to update our models much more frequently, with less effort, and to use more representative workloads in simulation experiments.

The two key challenges are to find a representation of traffic that supports the comparison of very different applications, and to understand the range of application behavior that exist in today's Internet. We address these challenges by representating traffic with an abstract model of network communication, and by exploring the structure of traffic using statistical cluster analysis. Our abstract model, described in Section 2, provides a very general representation of traffic that is application-independent but captures the nature of traffic at the source-level (rather than at the packet level) as a sequence of data exchanges between the two end points of a connection. The generality of this model makes it possible to compute a set of meaningful statistics (*features*) for each connection and to apply statistical clustering to group connections into a small number of *traffic clusters*. These traffic clusters group together connec-

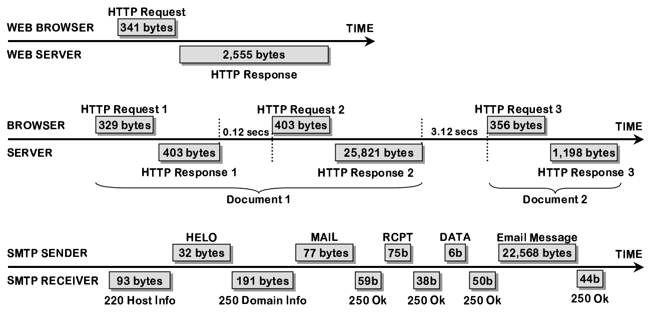

Figure 1: *Examples of application-level data exchanges. From top to bottom, non-persistent HTTP connection, persistent HTTP connection and SMTP connection.*

tions with similar source-level behavior, and their study helps to understand the most important communication strategies that are used by Internet applications. Our approach is described in Section 3. We also explore the structure of the traffic at the University of North Carolina and at Internet2 in two case studies that are presented in Section 4. Our efforts demonstrate that an important shortcoming of traditional clustering approaches is the difficulty of interpreting the results in a way that is meaningful for a network researcher. We overcame this problem by adapting to traffic analysis visualization techniques developed for the study of genetic data (in particular, gene expression arrays, a.k.a. microarrays). The idea is to combine the traditional hierarchical visualization (known as a *dendrogram*) and a heat map of connection features, making it possible to easily interpret the clustering result in terms of traffic types. This technique is demonstrated in our second case study. The later two sections of this paper present a (necessarily) brief overview of the related work and some conclusions.

2 Abstract Communication Model

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, 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 a 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). This representation is suitable for traffic generation at the source-level, in a close-loop fashion. The traffic generator would 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 *a-b-t model*, each point-to-point communication can be represented as a *connection vector* $(c_1, \ldots, c_n)$ with n *epochs*. An epoch is a triplet of the form $c_j = (a_j, b_j, t_j)$ that describe the data (a_j, b_j) and quiet time (t_j) parameters of the j'th exchange in a connection. 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* of $\tau = 1$ second, since we are only interested in capturing inter-epoch times that are clearly network-independent (*i.e.*, caused by application and user behavior and not by network dynamics). Notice that in the example above, the third and last quiet time 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 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 in Figure 1 correspond to applications that use TCP as their underlying process-to-process communication protocol. TCP's control data

[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.

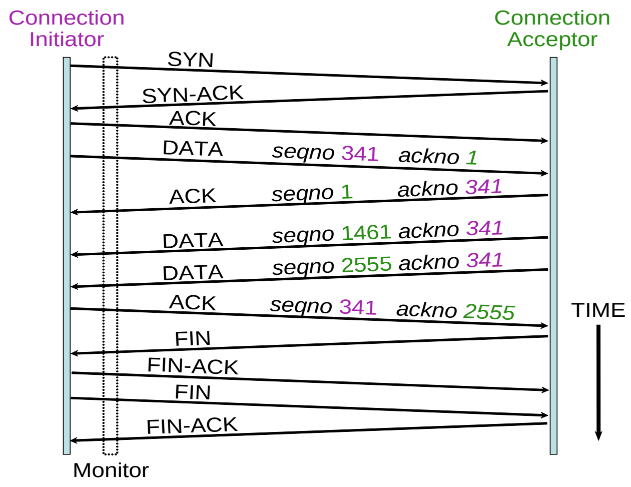

Figure 2: *TCP sequence numbers provide enough information to determine the size of Application Data Unit precisely.*

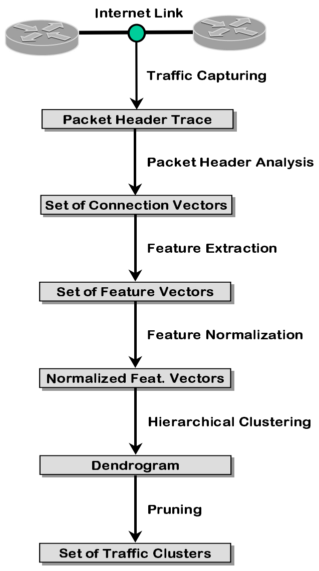

Figure 3: *Overview of our approach for clustering Internet communication patterns.*

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, or study the effect of the wireless medium of common traffic patterns.

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 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. The basic idea is shown in Figure 2 where a possible packet diagram for the first sample connection in Figure 1 is shown. ADU sizes can be measured using TCP sequence and acknowledgement numbers. The fundamental idea of our measurement algorithm is to reconstruct the logical order of segments in each TCP connection. This sorting makes it possible to accurately and efficiently measure ADU sizes in the presence of arbitrary packet reordering and retransmissions. In the example the first data unit is between sequence numbers 0 and 341 and the second one between 0 and 2555. The loss of the data segment with sequence number 1461 would imply the observation of a retransmission, if the segment is loss after the monitor, or reordering, if the segment is loss before the monitor. In either case, reordering the segments according to their logical order (*e.g.*, seqno 1461 goes before seqno 2555 and after seqno 341) makes it easy to measure ADU sizes by walking through ordered list of data segments and using changes in directionality of the data flow to detect ADU boundaries (*e.g.*, sequence numbers start increasing in a different direction for seqno 1461).

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

[2]This version of the model only deals with sequencial connections. Concurrent ones, in which at least two data units are sent in opposite directions at the same time, are not discussed here for brevity. Also, our current results show that concurrent communication is far less common than sequential one.

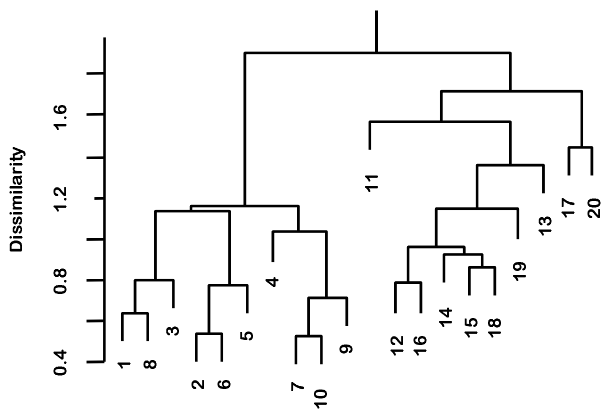

Figure 4: *Result of clustering a training set of 20 connections using agglomerative hierarchical clustering. Leaves labeled from 1 to 10 correspond to Telnet connections, while those labeled from 11 to 20 correspond to HTTP connections.*

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 beneficial 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 about the effectiveness of new network mechanisms.

Statistical clustering techniques, see *e.g.*, [14, 17, 10], 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 $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \in \mathbb{R}^d$ 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 sent by the initiator, the homogeneity in the sizes of the data units, and so on. Each feature is appropriately normalized so that its values lie between 0 and 1. We then measure the similarity between two connection vectors by the similarity between their associated feature vectors. We consider two alternative distance measures (see [10]), the standard Euclidean distance, *i.e.*,

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2},$$

and Pearson correlation coefficient, *i.e.*,

$$r_p(\mathbf{x}, \mathbf{y}) = \frac{S_{xy}}{\sqrt{S_x^2 S_y^2}}$$

where $S_{xy} = \sum_{i=1}^{d} (x_i - \overline{x})(y_i - \overline{y})$, $S_x^2 = \sum_{i=1}^{d} (x_i - \overline{x})^2$, $S_y^2 = \sum_{i=1}^{d} (y_i - \overline{y})^2$, and $\overline{x}$ and $\overline{x}$ are the mean values of $\mathbf{x}$ and $\mathbf{y}$ respectively. 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 in gene expression and other applications, and their graphical representation as trees (dendrograms) provides a useful way of identifying and analyzing groups of related communication patterns. Figure 3 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 described by this vector. 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 $a_{max} = \max\{a_j \in A\}$ 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. Given that inter-epoch times are more likely to reflect network properties, rather than only application-level behavior, we restrict our attention to a few time features: 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 more diverse and meaningful spectrum of values across different connections. Thus all correlation measurements are based on Spearman's rank correlation coefficient,

$$r_s = \sum_{i=1}^{d} (R_i S_i - u_d) v_d^{-1},$$

where $u_d = d(d+1)^2/4$ and $v_d = d(d^2-1)/12$, and R_i, S_i are the rank of x_i, y_i among $x_1, \ldots, x_d$ and $y_1, \ldots, y_d$

respectively. This is the non-parametric equivalent of Pearson's correlation 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.

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 greater than 1 or less than 0. Allowing features to take values outside the unit interval avoids the possible compression of their true 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 4 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 [16], 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 persistent (HTTP 1.1) 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. This visualization is a binary tree in which internal nodes represents split of the set of connections (with a y-axis height that correspond to the dissimilarity between its children). Leaves represent individual connections. The first split in the example cleanly separates Telnet connection from Web ones.

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 $(a^{+}_{max})/(a^{+}_{min}))$
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 R. C. with Lag 1

Table 1: *The 26 statistical features used in the divisive hierarchical clustering example shown in Figure 5.*

4 Clustering Examples

4.1 Divisive Hierarchical Clustering Example

In our first example of clustering traffic, we study a packet header trace collected during April 2002 at the main network link that connects the University of North 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 1. After normalizing the feature vectors, we analyzed them using the `diana` procedure with Euclidean distance as implemented R [16]. This algorithm is described in [19], and its basic idea is to sequentially split the cluster with the largest diameter by finding its most dissimilar observation. This observation is used as the seed of a new cluster, which will be populated with some number of similar observations from the original cluster.

The result of clustering the set of 5,000 are shown in Figure 5, 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 dendrogram reveals some useful structure in the set of connection. Connections in cluster 1 mostly cor-

[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.

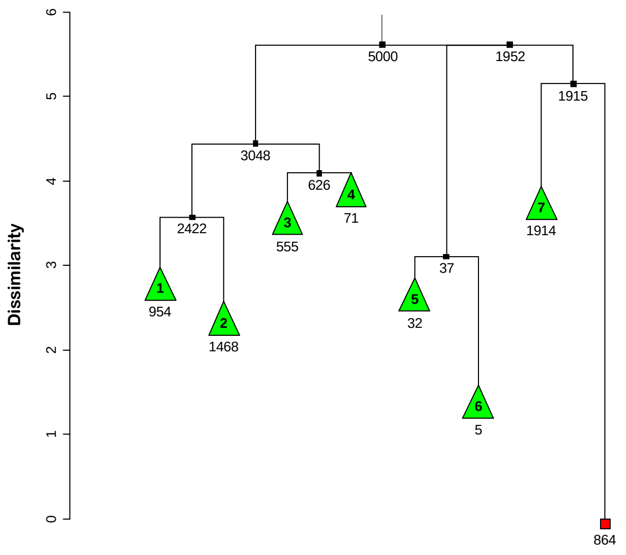

Figure 5: *Dendrogram obtained from the divisive hierarchical clustering of data set of 5,000 connections, pruned at depth 4.*

respond to HTTP, HTTPS (encrypted web traffic) and AOL traffic, while those in cluster 3 correspond to mail transfer protocols, such as SMTP and the Post Office Protocol (POP). The composition of clusters 2 and 4 was not so clear. 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 revealed 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 [21]. The sample consisted of 717 TCP connections[4]. Each connection was first transformed into a connection vector, and then summarized into a feature vector. Half of these connections were a random sample of port 80 connections, while the other half were a random sample of connection in other ports. The result was a matrix of 717 rows and 14 columns. Table 2 describes the 14 statistical features that were part of each vector.

[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.

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 $(a^+_{max})/(a^+_{min}))$
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 R. C. with Lag 1

Table 2: *The 14 statistical features used in the agglomerative hierarchical clustering in Figure 6.*

Feature vectors were clustered using the average-linkage agglomerative method proposed by Sokal and Michener [24], 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) [7]. The result of the clustering is shown in Figure 6. 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 (in other words, the larger the value, the lighter the color). On the left side of the array, a rotated dendrogram displays the hierarchical clustering of connections. 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. Note that this interpretation is based on port numbers (that we know are not very accurate), and it is only meant to illustrate the power of the method. 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

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

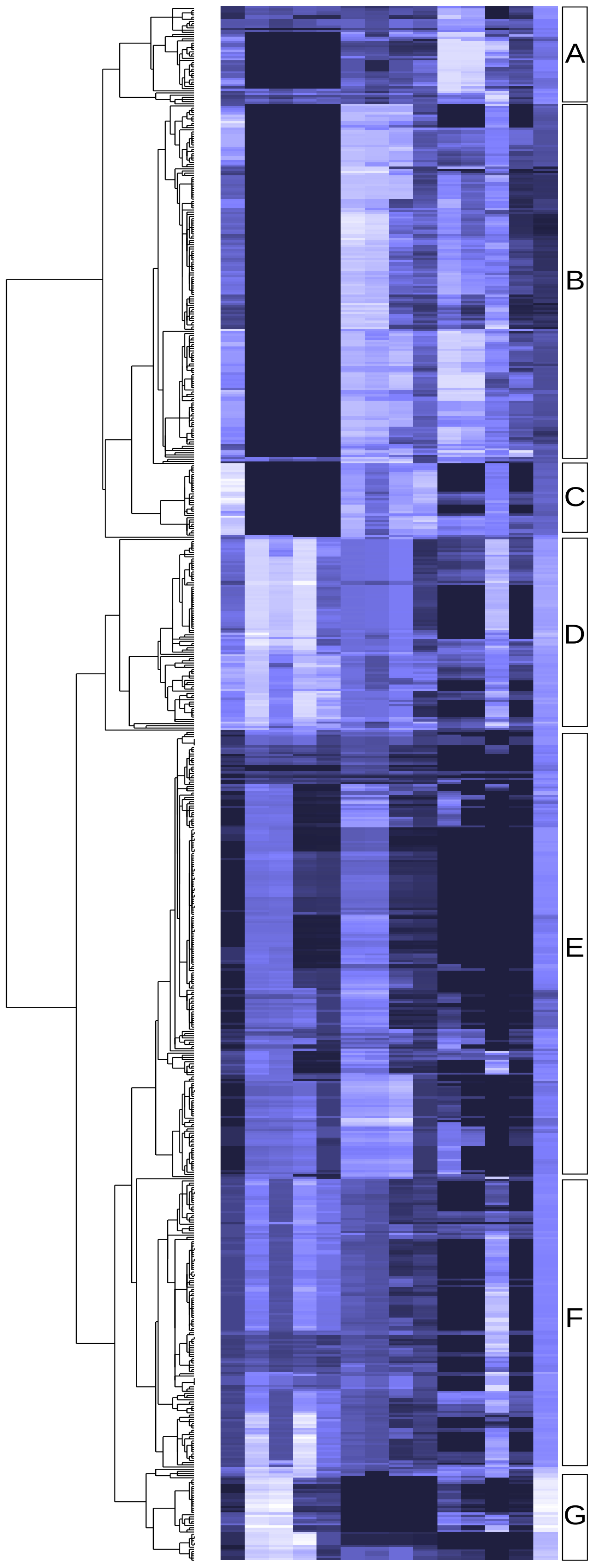

Figure 6: *Result of clustering a sample of connections from the Abilene-I data set. From left to right, the columns of the heat map correspond to* $n, a_{tot}, a_{2q}, a_{fd}, a_h, b_{tot}, b_{2q}, b_{fd}, b_h, t_{tot}, t_{2q}, dir, \rho_1$ and ρ_2.

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

Measuring and modeling traffic at the source-level has been an active area of research over the last ten years. 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.* [6, 2] and by Paxson [22]. Web traffic has been studied in numerous papers (*e.g.*, [20, 5, 15]), and file-sharing applications are the focus of much current work (*e.g.*, [13, 23]).

Traffic classification is known to be a difficult problem that has not received much attention in the past. There are a number of papers (*e.g.*, [18, 9]) that study how to identify groups of traffic that are *remarkable*, *e.g.*, consume a large fraction of the traffic, but their focus is not on understanding the source-level structure of traffic. Other relevant papers evaluate existing monitoring techniques and propose more powerful alternatives (*e.g.*, [8]).

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]. Prominent examples of research in traffic generation are Danzing *et al.* `tcplib` [6], the work by Barford and Crovella on web workload generation [1], and the SAMAN project by Lan and Heidemann [3].

6 Conclusion

We presented an abstract model of Internet communication and developed a methodology for clustering connections into a set of small groups. The use of recently developed visualization techniques makes it possible to easily interpret clustering results. We believe this provides a good starting points for understanding the types of source-level behaviors, and study how they change over time and across different vantage points. We are currently working on refining our measurement techniques and systematically examining the clustering structure of the traffic mixes at a large number of sites. We are also developing a new traffic generation tool that makes use of this structure to enable flexible traffic generation that is more representative of the wide variety of traffic found on the Internet today.

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 and Jeff Terrell for their 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] Kun chan Lan and John Heidemann. Rapid model parameteration from traffic measurement. Technical Report 561, USC/Information Sciences Institute, August 2002.

[4] Internet 2 Consortium. Internet2 Netflow Weekly Report, `http://netflow.internet2.edu/weekly/20040621`, June 2004.

[5] 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.

[6] 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.

[7] 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.

[8] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a better netflow. In *Proc. of the ACM SIGCOMM*, 2004.

[9] C. Estan, S. Savage, and George Varghese. Automatically inferring patterns of resource consumption in network traffic. In *Proc. of the ACM SIGCOMM*, 2003.

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

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

[12] Sprint Advanced Technology Laboratory (IP Group). IP monitoring project: Data management system, 2004. `http://ipmon.sprintlabs.com`.

[13] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP-19)*, 2003.

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

[15] F. Hernández-Campos, F. D. Smith, and K. Jeffay. Tracking the evolution of web traffic: 1995-2003. In *Proc. of the 11th IEEE/ACM Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2004)*, October 2003.

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

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

[18] B. Krishnamurthy and J. Wang. Traffic classification for application specific peering. In *Proc. of the ACM Internet Measurement Workshop*, 2002.

[19] 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.

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

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

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

[23] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proc. of the ACM SIGCOMM*, 2004.

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