The Extensive Challenges of Internet Application Measurement

Martin Arlitt Carey Williamson
Department of Computer Science
University of Calgary
E-mail: {arlitt,carey}@cpsc.ucalgary.ca

Abstract

The Internet has grown dramatically and evolved significantly over the past 10 years. While this growth provides new opportunities and many value-added services for millions of Internet users, this growth also presents ongoing challenges to networking researchers, for whom the study of the Internet itself is the *raison d'être*. There are many technical challenges other than the network link speed that make the monitoring of Internet applications difficult. To illustrate this point, this article presents several practical examples drawn from our experiences in Internet application measurement and monitoring.

Keywords: Internet Application Measurement

1 Introduction

This article describes some of the technical challenges that arise when measuring and monitoring Internet applications. The purpose of this article is to inform other networking researchers about the difficulties to expect and the pitfalls that can occur when measuring Internet applications on their own networks.

The first and perhaps most obvious challenge is finding a way to obtain data about Internet application usage. While there are numerous methods for obtaining data (some alternatives are discussed in Section 3), this article assumes that network packet traces will be collected. This technique enables the off-line analysis of Internet application usage.

Even after data collection, numerous technical challenges can arise. Examples of these additional technical challenges include:

- the evolution of Internet applications,
- applications that obfuscate their identity,
- incorrect protocol implementations,
- inappropriate use of protocol features, and
- the proliferation of application-level devices throughout the network.

The remainder of this article assumes that off-line analysis of network traces is used for measuring and analyzing Internet applications. The main focus in the rest of the article is on the five technical challenges listed above.

For readers interested in learning more about Internet measurement, the book by Crovella and Krishnamurthy [1] is a good starting point. This book provides a much broader and deeper treatment of Internet measurement than is possible in this article alone. Another useful resource is Paxson's article [2] on the "best practices" for Internet measurement.

The remainder of the article is organized as follows. Section 2 provides background information relevant for understanding how Internet applications are measured. Section 3 discusses different techniques for characterizing Internet applications. Section 4 describes in more detail the technical challenges that complicate the measurement of Internet applications. Section 5 concludes the article with a summary of our findings, and our observations on future challenges.

2 Background

This section provides a brief introduction to computer communications. This information provides the context required to understand the techniques used to measure Internet applications.

The left-hand side of Figure 1 shows a four-layer network reference model for the TCP/IP protocol suite [3]. This model defines a common framework for interconnecting computer systems. Each protocol layer in the model performs a specific task, offering a particular service to the higher layers. The higher layers can then use this functionality without having to know how that functionality is implemented. This layering concept is a major reason why the Internet has been so successful. Users do not need to understand the complex operational details of the Internet in order to use it; they only need to know how to use an Internet application, such as a Web browser.

TCP/IP Protocol Suite	Internet Examples	Type of Information Provided by Each Layer
Application	HTTP	HTTP request/response headers
Transport	TCP	source and destination port numbers; sequence numbers; acknowledgement numbers
Network	IP	source and destination IP addresses
Link	Ethernet	MAC addresses

Figure 1: The TCP/IP Protocol Stack

The middle column of Figure 1 shows some of the common protocols used in the Internet today. This example is for a packet generated by a Web browser as part of a client-server Web request/response transaction. In this particular example, four different protocols are shown: the HyperText Transfer Protocol (HTTP) at the application layer, the Transmission Control Protocol

(TCP) at the transport layer, the Internet Protocol (IP) at the network layer, and Ethernet (a popular local area networking technology) at the link layer.

The right-hand side of Figure 1 indicates the type of information that each layer provides. For example:

- the HTTP request and response headers provide information such as the name, size, and type of the Web object that was requested,
- the (well-known) TCP port numbers can indicate which Internet application is in use (e.g., TCP port 80 for the HTTP protocol in this example),
- the IP addresses (typically) indicate which host on the network sent the packet, and which host is the intended recipient, and
- Ethernet MAC addresses indicate the actual network devices sending and receiving the packet.

Throughout the remainder of the article, the discussion focuses primarily on the application-layer and transport-layer protocols.

3 Collecting Data on Internet Application Usage

A useful tool (or tool suite) for characterizing Internet applications should perform the following two functions. First, it must be capable of identifying each distinct application on the network, or at least those that contribute significantly to the traffic volume. Second, the tool must be able to extract information on important characteristics of the application. For example, if the tool showed that Web traffic was a significant application on the network under study, one might require information for evaluating the effectiveness of a Web caching architecture. This would require (among other things) extracting relevant headers from the HTTP requests and responses.

There are two general categories of network measurement techniques: *active* and *passive*. Active techniques inject packets into the network in order to measure a specific characteristic. For example, `ping` is often used to measure the round-trip time between two hosts. Passive techniques, on the other hand, only observe traffic that is already on the network, without contributing any traffic of their own. While both categories have their advantages and disadvantages, we focus on passive techniques, since they are more commonly used for characterizing Internet applications.

A typical approach for characterizing Internet applications is to analyze traffic on Internet links. This leads to another challenge, namely the speed of the network links. For example, current Internet backbone links of 10 Gb/s or greater can carry peak loads of millions of packets per second. Simply keeping statistics at such rates is a challenge [4]. An alternative approach is to capture packet traces of the network activity, and then analyze the traces off-line. However, recording packet-level information can also be a challenge.

There are several well-known and widely-used techniques for collecting packet traces for offline analysis of Internet usage. In most techniques, a fundamental tradeoff exists between two issues:

- how detailed is the information collected, and

- how scalable is the approach.

"Scalability" refers to the general efficiency of an approach, and its ability to perform adequately as the problem size or state space is increased. For example, Internet traffic measurement requires approaches that are scalable with respect to traffic volume, network link speed, and the number of protocols, applications, and users that need to be monitored.

Scalability is generally achieved by reducing the number of packets that must be processed, and by reducing the per-packet overhead for processing and archiving. However, these strategies typically result in the loss of information.

In this section, we discuss different measurement techniques for characterizing Internet applications. The techniques are ordered from the most detailed to the least detailed (or equivalently, from the least scalable to the most scalable). Each of these techniques is applicable in certain situations, though it is always important to know the strengths and limitations of a particular technique.

3.1 Complete Packet-Level Traces

The most obvious approach to network traffic measurement and monitoring is the collection and analysis of complete packet-level traces from an operational network. These traces can be collected using special-purpose equipment such as a commercial network traffic analyzer, or public-domain network monitoring software such as `tcpdump` (`http://www.tcpdump.org/`) or `ethereal` (`http://www.ethereal.com/`). A complete packet-level trace records all of the information in all of the TCP/IP packets traversing the network being monitored.

This technique enables a complete view of Internet application characteristics. This capability comes from the collection of complete packet payload information, as well as the packet headers, enabling an in-depth analysis of user, application, protocol, and network behaviours.

Complete packet-level trace collection has some significant disadvantages. In particular, the detailed information collected limits the scalability of the technique. First, capturing lengthy traces of full data packets requires significant storage capacity. Second, the passive monitoring tool has no method for throttling the rate at which clients and servers exchange data [5]. If the tool misses some packets, it reduces the completeness of the trace, which will ultimately reduce the accuracy of the ensuing analysis.

3.2 Trace Reduction Techniques

There are several well-known approaches for improving the scalability of network measurement techniques. The first of these is to examine only the packets that are associated with a particular application. This is typically done based on well-known TCP port numbers (e.g., port 80 for Web traffic). A second technique is to minimize the amount of data that is archived. This may involve retaining only a summary of each transaction, retaining only the application-level request and response headers (and not the entire Web page payload), or retaining only aggregated information. A third technique is to capture only TCP/IP packet headers, rather than complete packet payloads. This approach can significantly reduce the per-packet processing overhead, although at a loss of application-layer information. This technique has been used in many studies, including that by Cáceres *et al.* [6]. All of these techniques can help address other issues, such as privacy concerns.

3.3 Analysis of SYN/FIN/RST Packets

One of the first longitudinal studies of Internet traffic behaviour was presented by Paxson [7]. His work showed that long-term analysis of TCP/IP traffic behaviour is possible by recording packet-level information only for the TCP SYN, FIN, and RST packets on the network. The SYN (Synchronize) flag is used in the opening handshake for a new TCP connection, the FIN (Finish) flag is used in the closing handshake for a connection, and the RST (Reset) flag is used to abort an existing connection.

These TCP control packets indicate not only the start and end of the TCP connection, but also the cumulative volume of data transmitted in each direction. These properties come from the time stamp associated with each packet, as well as the starting and ending sequence numbers that appear in SYN and FIN packets, respectively. In other words, using the SYN and FIN packets, one can typically determine characteristics such as the participating hosts, the Internet application, the start time of the connection, the TCP connection duration, the number of bytes transferred in each direction, and the average connection throughput.

The advantage of this technique is that the SYN/FIN control packets constitute only a small proportion of the total packet volume on the network, yet they provide numerous (but not all) application-level traffic characteristics. However, this technique cannot provide information on application interactions with the transport layer, such as the number of packets in the connection, the packet sizes used, or the packet arrival process (indicative of TCP's congestion control mechanisms), nor can it reveal application-level protocol information such as the request and response headers used, the number of transactions per connection, and the size of individual transactions.

In principle, RST packets can provide the same information as FIN packets. However, Paxson found that the sequence numbers in RST packets are often meaningless [7]. As a result, RST connections are usually ignored in traffic characterization studies.

3.4 Unidirectional Traffic Analysis

TCP is a full-duplex protocol, which allows the simultaneous exchange of data and acknowledgements in each direction of a TCP connection. The discussion of measurement techniques in the foregoing sections has assumed that network monitoring is being done for *both* directions of a TCP traffic flow, which may be on different physical links. Bi-directional traffic analysis seems like the natural (and perhaps the only) approach to the problem.

In 2001, Smith *et al.* [8] demonstrated the feasibility and usefulness of collecting and analyzing *unidirectional* traces of TCP/IP packet headers. The essence of the approach is to use the advancement of sequence numbers to determine data flow in one direction, and the advancement of the acknowledgement numbers to determine data flow in the other direction. This technique applies regardless of whether the network monitor is seeing the client side or the server side of a connection. Similar techniques are often used in bidirectional TCP analysis, but unidirectional analysis explicitly relies upon the full-duplex structure of TCP data transfers.

The primary advantage of the unidirectional analysis technique compared to bidirectional analysis is a reduction in the number of packets that have to be processed (about 50% on average, depending on the TCP acknowledgement strategy being used and the traffic direction being analyzed). Furthermore, the technique provides most of the information available from the SYN/FIN/RST technique, while providing even more information on selected application-level protocol characteristics. For example, Smith *et al.* show how increases in the client data sequence numbers indicate persistent-HTTP usage, and thus can reveal how many HTTP transactions use persistent TCP connections. This technique can also reveal the number and size of objects that were requested and received on each persistent connection. A disadvantage of this technique is that it cannot provide information such as the application-level request and response headers.

3.5 Packet and Flow Sampling

Another technique for improving the scalability of network measurement is *packet sampling.* The general concept is to capture and analyze one out of every N packets, rather than every packet. Packet sampling techniques can provide a range of information, including distributions of packet sizes, packet types, and inter-arrival times. Although packet sampling techniques can provide some application-level information, the usefulness of this information will depend on the particular sampling used. Furthermore, most sampling techniques are not useful if reconstructing complete application-layer transactions is required.

There are many approaches for identifying which packets to select. One approach involves using randomness in the sampling process, to prevent synchronization with periodic processes in the traffic. This approach is recommended by `sflow.org` (`http://www.sflow.org/`), which is an international multi-vendor end-user forum. `sflow` relies on sampling for measuring and monitoring traffic in complex networks.

Flow sampling is an approach that can be used instead of or in addition to packet sampling to further improve the scalability of network measurement. For example, Duffield *et al.* developed *threshold sampling* [9]; this technique records statistics on flows that exceed a certain threshold (e.g., more than N packets). This improves the accuracy of the reported flow statistics compared to simpler (e.g., uniform) sampling techniques.

4 Technical Challenges

The past decade has seen tremendous advances in technology. Processor speeds, network link bandwidths, and storage capacities have all increased by more than an order of magnitude. This period of time has also seen the development of some impressive network measurement tools, including commercially-available network cards that can capture packet traces at 10 Gb/s rates. However, collecting detailed application-level traces in a continuous manner remains a challenge. As previously mentioned, the primary issue is that network link bandwidths continue to increase at least as fast as processor speeds and storage capacities. As a result, bottlenecks such as memory or I/O bus bandwidth can limit the performance of a network monitor. As mentioned in Section 3, there are two approaches to improving the scalability of a passive network measurement tool: reduce the number of packets that are processed, and minimize the per-packet overhead for processing and archiving. These issues have received considerable attention for the purpose of network planning and administration. For example, Zhao *et al.* propose several data streaming algorithms for creating accurate and efficient traffic and flow matrices [10]. Such techniques can also be used to identify traffic anomalies such as Distributed Denial of Service (DDoS) attacks. Schweller *et al.* developed *reversible sketches* [11], which can preserve information such as source IP addresses (crucial for identifying sources of heavy flows) without retaining per-flow state. Their prototype implementation achieved a throughput of 16 Gb/s, demonstrating the scalability of their approach [12]. These data streaming techniques, however, do not address other technical challenges in identifying Internet applications.

In the past ten years, our involvement with characterizing Internet applications has identified numerous technical challenges beyond the fundamental issue of network data rates. In the remainder of this section we discuss several of these challenges in more depth.

4.1 Evolving Internet Applications

On of the challenges is the evolution of Internet applications, which motivates the continued characterization of such applications over time. For example, a decade ago the Web was just emerging as a significant application. At that time, most TCP connections transferred only a single Web file. This approach is often inefficient, because of the extra round trip times incurred for TCP connection establishment and release, as well as the initial slow start undertaken by each new connection. This observation led to the development of a more efficient technique called persistent connections, which is now part of HTTP/1.1 [13]. This evolutionary change altered the behaviour of Web transactions observed on the Internet. Similarly, the evolving mix of applications is one of the challenges for network simulation studies [14].

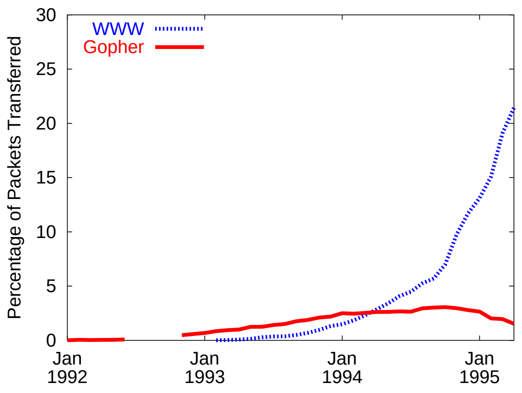


Figure 2: Evolution of WWW and Gopher, 1992-1995

Figure 2 provides further motivation for the systematic characterization of Internet applications over time. Figure 2 shows the percentages of packets transferred over the NSFNET backbone between 1992 and 1995 (`ftp://nic.merit.edu/nsfnet/statistics/`) that were attributable to the World Wide Web (WWW) and Gopher applications (note that NSFNET statistics by port number were not available for several months in 1992 during the transition to the T3 backbone.)

WWW and Gopher both emerged as new Internet applications in the early 1990s. The primary attraction to these applications was a simpler interface than existing applications for accessing information on the Internet. Initially, Gopher accounted for much more traffic than WWW. However, as Figure 2 indicates, the WWW soon became a popular Internet application, while Gopher faded into obscurity. This information clearly indicated that by 1995 researchers should be focusing on improvements to the WWW application rather than Gopher. Having similar information today and in the future would indicate whether continued WWW research is needed, or if other applications now require our attention.

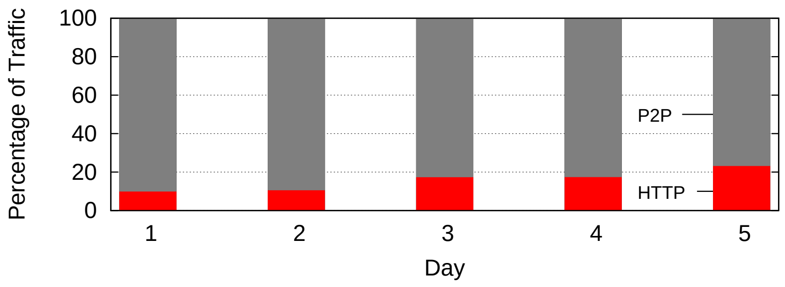
Peer-to-Peer (P2P) file sharing is a more recent Internet application. Many P2P networks have emerged, and P2P quickly became one of the dominant applications on many networks. Initially these applications were easy to identify, as they used well-known TCP port numbers, just like other Internet applications. However, several of these applications evolved rapidly, primarily to elude network operators attempting to block P2P traffic using port-based firewall rules.

One significant change occurred in September 2002, when the KaZaA 2.0 desktop version was released. One of the new features of this version was the use of dynamically-assigned TCP port numbers. That is, KaZaA stopped relying on well-known port numbers to exchange data. This change made it difficult for network operators (and researchers) to identify KaZaA traffic [15].

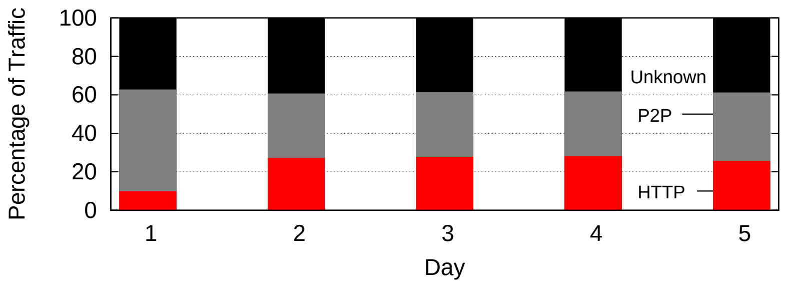
Figure 3 shows an example of how significant this single change was. Figure 3(a) shows the breakdown of traffic on a commercial Internet link for five days in September 2002. According to port-based classification, 10-25% of the byte traffic volume was HTTP, with the remaining 75-90% from known P2P applications. Figure 3(b) reveals some significant changes. By the beginning of October, about 40% of the byte traffic volume on the same Internet link was not identifiable using port-based classification. Since October 2002, port-based classification on Internet links has essentially become useless, as the majority of application traffic can no longer be classified with this technique.

An emerging challenge to Internet application measurement is encryption. There are now freely available encryption libraries that applications can readily use. As computing platforms become more powerful, encrypting all data transmissions at the application, transport, or network layer will become feasible. Some applications (e.g., BitTorrent) already support encryption, while others (e.g., Skype) already encrypt all communication. Once encrypted communication becomes prevalent, our understanding of Internet application behaviour (at least with current measurement techniques) is likely to diminish further.

The rapid evolution of applications in the past few years has led numerous networking researchers to propose new classification techniques. Two general techniques are followed: inspecting packet payloads (e.g., [16]), or analyzing transport-layer characteristics (e.g., [17]). Payload inspection techniques work well if application signatures can be determined, but are computationally intensive and do not work if the payloads are encrypted. Classification techniques based on transport-layer characteristics are not vulnerable to payload encryption, but may require more manual effort for labeling applications.



(a)



(b)

Figure 3: Evolution of Peer-to-Peer Traffic: (a) September 2002; (b) October 2002

4.2 Application Masquerading

An underlying assumption of using well-known port numbers for Internet application classification is that no other applications use that port. In the past, there was little evidence to suggest this assumption was being violated.

The strength of this assumption has weakened over time, however, particularly since the emergence of P2P file sharing. For example, P2P applications such as KaZaA provide their peers with the ability to communicate with other peers on port 80. The simplest way to distinguish between a Web application and a KaZaA application (both using TCP port 80) is to examine the application-level information, as shown in Figure 4. This cannot be done by some of the measurement techniques described in Section 3.

Figure 4(a) provides an example of a typical Web transaction. The transaction begins with a Web client issuing an HTTP request for a desired Web object ("GET /index.htm"). The initial line of the server's response ("HTTP/1.1 200 OK") indicates that the request was successful. The server response then includes a number of HTTP headers, including `Date, Server, Last-Modified` and `Content-Length`. Finally, the response includes the requested object.

Figure 4(b) shows an example of a masquerading application. This particular example is a KaZaA data transfer that occurred over TCP port 80. In this example, KaZaA uses HTTP to transfer files, so the format of the request and several of the response headers is the same as for the Web transaction. However, this can be recognized as a KaZaA transaction from the `Server` type as well as the `X-Kazaa` headers. The port numbers listed in the `X-Kazaa-IP` and `X-Kazaa-SupernodeIP`

```
(a) World Wide Web (TCP port 80)
GET /index.htm HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 02 Feb 2004 19:28:35 GMT
Server: Apache/1.3.27 (Unix)
Last-Modified: Fri, 30 Jan 2004 20:53:03 GMT
ETag: "451288-34db-401ac42f"
Accept-Ranges: bytes
Content-Length: 13531
Connection: close
Content-Type: text/html

(requested object)
```

```
(b) KaZaA (TCP port 80)
GET /index.htm HTTP/1.0

HTTP/1.1 200 OK
Content-Length: 67
Accept-Ranges: bytes
Date: Mon, 02 Feb 2004 20:34:47 GMT
Server: KazaaClient Jul 6 2003 17:54:13
Connection: close
Last-Modified: Mon, 02 Feb 2004 20:33:00 GMT
X-Kazaa-Username: zzzzzzzzzz
X-Kazaa-Network: KaZaA
X-Kazaa-IP: 192.168.1.1:1156
X-Kazaa-SupernodeIP: 192.168.2.1:1055
X-KazaaTag: 3==hGORbERtzzj8RREfgrqqAbz///8=
Content-Type: application/octet-stream

(requested object)
```

Figure 4: Two TCP Port 80 Transactions: (a) WWW (b) KaZaA

headers (1156 and 1055, respectively) indicate the dynamically selected TCP ports that these two peers are listening on. The KaZaA peers can also be contacted at TCP ports 80 and 1214 (the well-known port for KaZaA).

4.3 Incorrect Protocol Implementations

When developing a tool to extract application protocol information, two obvious design principles are to optimize for the common case and to minimize the number of sanity checks that the tool performs. This strategy reduces the per-packet processing overhead, but at the risk of incorrectly interpreting a packet.

If all network protocol implementations are correct, then this risk is negligible. However, since such implementations are complex, it is possible for small errors to remain undiscovered for long periods of time. In this regard, network monitoring tools can be helpful as a debugging tool.

One example of an unusual error encountered is a vendor-specific problem, in which an IP-in-IP encapsulation was incorrectly marked as a regular TCP packet. Both the client and the server appeared to have this same implementation error, as the TCP connection in the encapsulated flow worked successfully. However, network monitoring tools might not interpret these packets properly. Good sanity checks are required to detect such problems. We were originally alerted to this anomaly because the TCP byte counts reported by our monitor were impossibly high.

Due to such scenarios, one should never assume that all protocol implementations on a heterogeneous network are correct. For most applications of network monitoring, accurate measurements are highly desirable. For others, such as usage-based billing, they are a requirement.

4.4 Inappropriate Use of Protocol Features

A common approach in network traffic analysis is to discard and ignore the packets from TCP connections that are reset. After all, the RST flag is supposed to indicate that something seriously wrong has happened with the TCP connection.

However, in a recent analysis of HTTP traffic [18], we discovered that approximately 20% of TCP connections carrying HTTP traffic are reset. Furthermore, these reset connections accounted for almost half of the total bytes transferred by all of the HTTP connections.[1]

Our investigation of this phenomenon identified two prominent sources of the TCP resets. First, some Web servers issue resets to close idle persistent connections. These servers do this in order to reclaim memory resources. Second, some Web browsers reset their idle connections. This permits the server to reclaim more quickly the memory resources used by the TCP connections.

These observations indicate that reset TCP connections must be included to understand Internet application characteristics properly. Any network measurement tool that ignores RST connections might be underestimating "normal" network usage by 20-50%.

4.5 Network Device Complexity

The past ten years have witnessed the proliferation of new devices within the network: firewalls, NAT boxes, packet shapers, Web caches, etc. Many of these devices can affect the mixture of application traffic seen by a network monitor. For example, firewalls may prevent certain applications from being used on the network; packet shapers may limit the throughput of selected applications, or even alter the TCP/IP packet headers; and Web caches may serve some fraction of Web requests locally, thus reducing the amount of Web traffic that leaves the network. As a consequence of these types of devices, the results of a study of Internet application behaviour on one network may not reflect the behaviour on other networks (with a different set of devices).

Another issue with network devices is that their growing functionality and complexity can lead to problems, due to implementation errors or anomalous protocol interactions. For example, some Web caches are capable of operating in a *transparent mode*. This means that users, and sometimes the (origin) servers, are unaware of the presence of the cache.

Transparent caches are appealing because they do not require user-level configuration changes to Web browsers. When a Web request is sent over the network, an "intelligent" switch redirects the request to the Web cache. In this way the Web cache appears as the server to the user. If the requested object is not in the cache, then the cache must retrieve the object from the actual origin server. Some Web caches operate completely transparently, in that they appear as the server to the user, and as the user to the server. However, this can be tricky to implement and configure correctly, and problematic to troubleshoot, as the following scenario demonstrates.

In a commercial network on which a transparent Web cache was deployed, some users were occasionally experiencing lengthy delays. Although these problems were initially thought to be related to either Internet delays or load on the origin server, a close inspection of the Web cache's

[1]Smith *et al.* reported that 15% of HTTP connections transferred multiple objects, accounting for 40% of the total bytes transferred. They did not report the RST behaviour though.

operation revealed the actual root cause. For the Web cache in question, each time a requested object was not found in the cache, the cache would establish a TCP connection to the origin server to retrieve a copy of the object. In order for the cache to be completely transparent, the cache utilized the user's IP address as its own, and relied on the intelligent switch to properly distinguish between packets destined to the user's computer, and those that should be redirected to the cache. On a miss, this particular cache chose a monotonically increasing port number when initiating a TCP connection with the origin server. The resulting connection tuple (user's IP address, cache's selected port number, origin server's IP address, origin server's port number) occasionally conflicted with an already established connection between the user's computer and the transparent cache. When this occurred, the "intelligent" switch would incorrectly handle the packets for one of the connections, causing the unlucky connection to hang indefinitely (until the affected user became frustrated and aborted their connection). Under peak load this problem occurred occasionally yet repeatedly, degrading the Web browsing experience for numerous users.

These complicated protocol interactions between devices are an issue for network monitoring as well. First, they require that the monitor include a robust network protocol stack. This makes the monitor more complicated to implement. Second, the monitoring resources consumed by the failed transactions reduce those available for properly functioning transactions. Both of these problems reduce the performance and scalability of the tool.

5 Summary and Conclusions

In the past decade the Internet has changed significantly. There are more users, businesses, and organizations on-line. Improved technologies have allowed new applications to emerge. Unfortunately, we have not been able to measure these changes systematically, for several reasons.

In this article we explored some of the technical challenges for measuring Internet applications. While the most obvious challenge continues to be the speed of network links (relative to processing speeds and storage capacities), there are a number of other important issues that either reduce the accuracy or scalability of all existing network traffic measurement techniques. These issues include:

- Internet application evolution
- application masquerading
- incorrect protocol implementations
- inappropriate use of protocol features
- the growing complexity of inter-networking devices.

Some of these challenges are related to human error, and are thus likely to persist for the foreseeable future. Others are related to human behaviour and the nature of the Internet, and we expect that these challenges will also endure. Both categories pose challenges for network operators and networking researchers alike.

Acknowledgements

Financial support for this work was provided by iCORE (Informatics Circle of Research Excellence) in the Province of Alberta, and by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] M. Crovella and B. Krishnamurthy, *Internet Measurement: Infrastructure, Traffic, and Applications*, John Wiley & Sons, Ltd., West Sussex, UK, 2006.

[2] V. Paxson, "Strategies for Sound Internet Measurement", *Proceedings of 2004 ACM SIGCOMM Internet Measurement Conference*, pp. 263-271, Taormina, Italy, October 2004.

[3] R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, Reading, MA, 1994.

[4] S. Ramabhadran and G. Varghese, "Efficient Implementation of a Statistics Counter Architecture", *Proceedings of ACM SIGMETRICS 2003 Conference*, pp. 261-271, San Diego, CA, June 2003.

[5] A. Feldmann, "BLT: Bi-Layer Tracing of HTTP and TCP/IP", *Computer Networks*, vol. 33, no. 1-6, pp. 321-335, June 2000.

[6] R. Cáceres, P. Danzig, S. Jamin and D. Mitzel, "Characteristics of Wide-Area TCP/IP Conversations", *Proceedings of ACM SIGCOMM '91*, pp. 101-112, Zurich, Switzerland, 1991.

[7] V. Paxson, "Growth Trends in Wide-Area TCP Connections", *IEEE Network*, vol. 8, no. 4, pp. 8-17, July 1994.

[8] F. Smith, F. Campos, K. Jeffay and D. Ott, "What TCP/IP Protocol Headers Can Tell Us About the Web", *Proceedings of ACM SIGMETRICS 2001*, vol. 29, no. 1, pp. 245-256, Cambridge, MA, June 2001.

[9] N. Duffield, C. Lund and M. Thorup, "Learn More, Sample Less: control of volume and variance in network measurement," *IEEE Transactions in Information Theory*, Vol. 51, No. 5, pp. 1756-1775, 2005.

[10] Q. Zhao, A. Kumar, J. Wang and J. Xu, "Data Streaming Algorithms for Accurate and Efficient Measurement of Traffic and Flow Matrices", *Proceeding of ACM SIGMETRICS 2005*, vol. 33, no. 1, pp. 350-361, Banff, AB, June 2005.

[11] R. Schweller, A. Gupta, E. Parsons and Y. Chen, "Reversible Sketches for Efficient and Accurate Change Detection over Network Data Streams", *Proceedings of 2004 ACM SIGCOMM Internet Measurement Conference*, pp. 207-212, Taormina, Italy, October 2004.

[12] R. Schweller, Z. Li, Y. Chen, Y. Gao, A. Gupta, Y. Zhang, P. Dinda, M. Kao and G. Memik, "Reverse Hashing for High-speed Network Monitoring: Algorithms, Evaluation, and Applications", *Proceedings of IEEE Infocom 2006*, Barcelona, Spain, April 2006.

[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, HTTP Working Group, June 1999. `www.ietf.org/rfc/rfc2616.txt`

[14] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet", *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 392-403, August 2001.

[15] C. Colman, "What to Do About P2P?", *Network Computing Magazine*, Vol. 12, No. 6, November/December 2003. `http://www.networkcomputing.co.uk/articles/reviews.asp?a_id=35`

[16] J. Ma, K. Levchenko, C. Kreibich, S. Savage and G. Voelker, "Unexpected Means of Protocol Inference", *Proceedings of the 2006 ACM SIGCOMM Internet Measurement Conference*, pp. 313-325, Rio de Janeiro, Brazil, October 2006.

[17] A. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques", *Proceedings of ACM SIGMETRICS 2005*, vol. 33, no. 1, pp. 50-60, Banff, AB, June 2005.

[18] M. Arlitt and C. Williamson, "An Analysis of TCP Reset Behaviour on the Internet", *ACM Computer Communication Review*, Vol. 35, No. 1, pp. 37-44, January 2005.

TCP/IP Protocol Suite	Internet Examples	Type of Information Provided by Each Layer
Application	HTTP	HTTP request/response headers
Transport	TCP	source and destination port numbers; sequence numbers; acknowledgement numbers
Network	IP	source and destination IP addresses
Link	Ethernet	MAC addresses

Figure 1: The TCP/IP Protocol Stack

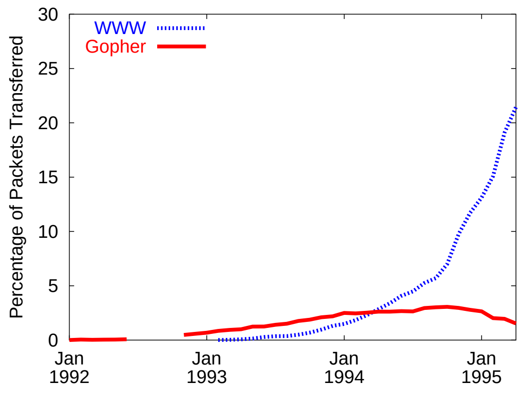
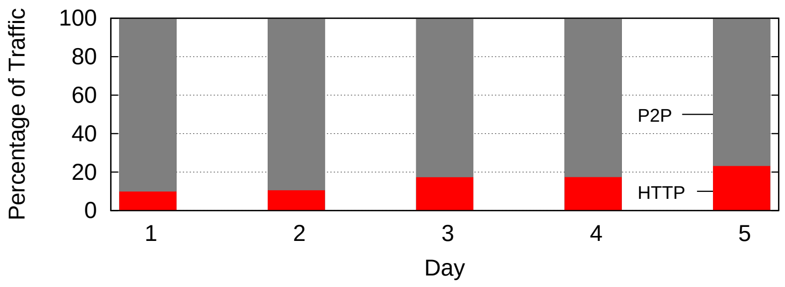
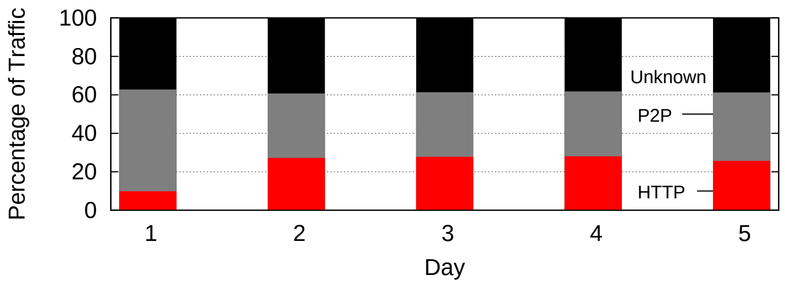


Figure 2: Evolution of WWW and Gopher, 1992-1995



(a)



(b)

Figure 3: Evolution of Peer-to-Peer Traffic: (a) September 2002; (b) October 2002

```
(a) World Wide Web (TCP port 80)
GET /index.htm HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 02 Feb 2004 19:28:35 GMT
Server: Apache/1.3.27 (Unix)
Last-Modified: Fri, 30 Jan 2004 20:53:03 GMT
ETag: "451288-34db-401ac42f"
Accept-Ranges: bytes
Content-Length: 13531
Connection: close
Content-Type: text/html

(requested object)
```

```
(b) KaZaA (TCP port 80)
GET /index.htm HTTP/1.0

HTTP/1.1 200 OK
Content-Length: 67
Accept-Ranges: bytes
Date: Mon, 02 Feb 2004 20:34:47 GMT
Server: KazaaClient Jul 6 2003 17:54:13
Connection: close
Last-Modified: Mon, 02 Feb 2004 20:33:00 GMT
X-Kazaa-Username: zzzzzzzzzz
X-Kazaa-Network: KaZaA
X-Kazaa-IP: 192.168.1.1:1156
X-Kazaa-SupernodeIP: 192.168.2.1:1055
X-KazaaTag: 3==hGORbERtzzj8RREfgrqqAbz///8=
Content-Type: application/octet-stream

(requested object)
```

Figure 4: Two TCP Port 80 Transactions: (a) WWW (b) KaZaA

Carey Williamson is an iCORE Chair in the Department of Computer Science at the University of Calgary, specializing in *Wireless Internet Traffic Modeling*. He holds a B.Sc.(Honours) in Computer Science from the University of Saskatchewan, and a Ph.D. in Computer Science from Stanford University. His research interests include Internet protocols, wireless networks, network traffic measurement, network simulation, and Web performance.

Martin Arlitt is a Senior Research Associate in the Department of Computer Science at the University of Calgary and a Senior Scientist at HP Labs. He has been a member of IEEE for 10 years. His research interests include measurement, characterization, and performance evaluation of computer systems and networks.