Looking for Science in the Art of Network Measurement

Matthias Grossglauser[1] and Balachander Krishnamurthy[1]

AT&T Labs - Research, 180 Park Ave, Florham Park NJ 07932, USA
{mgross,bala}@research.att.com

Abstract. Network measurements are crucial both to drive research and in network operations. We introduce a taxonomy and survey the state of the art of network measurement. We compare measurements available at the network layer with those available at higher layers, specifically the DNS and the Web. Both the DNS and the Web can be viewed as logical networks; this allows the direct comparison of measurement methods available at these layers with those available at the network layer. We argue that measurement support within the DNS and the Web is insufficient in light of the fact that they affect end-user performance as much as the network layer. We derive some recommendations for the reuse of network layer measurement methods in the DNS and the Web.

1 Introduction

The Internet has evolved into a system of astonishing scale and complexity, fraught with conflicting economic interests, comprised of subsystems from an unprecedented range of vendors, and burdened by many short-sighted fixes to fundamental problems (such as the deployment of NATs in response to a shortage in IP addresses). The research community has no hope of completely modeling the Internet [36]; only crude abstractions that focus on very specific subproblems are within our reach. Therefore, we are called upon to *discover* the behavior of the Internet, in addition to *modeling* aspects of it. The collection, analysis, and interpretation of measurements is key to this aspect of Internet research. It parallels other fields of study that are concerned with systems that escape exhaustive modeling, such as econometrics and biometrics. For example, measurements have revealed surprising statistical features in network traffic that were not predicted by any models before their discovery [27]; clearly, exploratory measurement studies deserve an important place in the research agenda.

Measurements are also crucial in the operation of the Internet. Traffic control and engineering, i.e., routing and resource allocation to make the best use of network resources while maximizing performance, directly depends on traffic measurements [9]. Other examples include accounting and billing, intrusion and attack detection, verification of service level agreements (SLAs), measurement-based admission control [20, 14, 18], etc.

At higher layers, measuring traffic at popular Web sites as well as examining problems like flash crowds [23, Chapter 11] (sudden surge in traffic aimed at a site) require measurements where typically administrative access is not available. Identifying the location of clients, mirroring popular sites and resources [28], improving the performance of individual servers, and examining ways to move popular services to edge of the network require large scale measurements.

In this paper, we give a brief survey of the state of the art of network measurement. For this purpose, it is not useful to think of the Internet is terms of the traditional layered model. Rather, we should think of the Internet as being composed of multiple subsystems that exhibit network structure themselves. Chief among these overlay networks are the domain name system (DNS) and the World Wide Web (consisting of clients, origin servers, proxies, caches). Other examples include content distribution networks (CDNs) and peer-to-peer networks, such as Napster [29] and Gnutella [15].

We argue that measurement efforts within these overlay networks face similar challenges and pitfalls as measurement efforts at the network layer, which are arguably more mature. To structure the comparison, we examine three classes of measurements for the network layer, for the DNS, and for the Web: topology, state, and traffic. The *topology* is the static, underlying structure of each network (e.g., physical links between routers, or the static configuration of a proxy in a browser). The *state* refers to dynamic changes in the active topology and other variables *not directly related to traffic* (for example, the utilization of a link would not be considered a state variable, while the operational state of a link is). The *traffic* refers to the flow of work through the network (e.g., packets through the IP network, name resolution requests and responses through the DNS network).

Our systematic comparison illustrates the fact that logical networks, which affect end user performance, are severely under-instrumented today. The visibility into the DNS and the Web is much more limited than at the network layer, with the result that troubleshooting, control and engineering is significantly more challenging for these networks. The purpose of this paper is to point out some of these shortcomings, and to suggest possible remedies inspired by measurement support at the network layer.

The paper is structured as follows. Section 2 briefly summarizes the state of the art in measurement at the network layer. Section 3.1 and section 3.2 give an analogous assessment for the DNS and the Web. We summarize our findings and proposed remedies in Section 4.

2 IP Network Measurements

In this section, we give an overview of measurements available at the IP network layer. For each type of measurement - topology, state, and traffic - we describe what measurements are available externally (i.e., without having administrative control over the network) and internally (with administrative access).

2.1 Topology

The topology of the Internet can thought of as having two levels of hierarchy: the autonomous system (AS) level and the network domain level. We mainly focus on the domain level.

External measurements Several classes of methods have been proposed to infer the topology of the Internet from external measurements. The first class of methods, commonly referred to as *topology discovery*, relies on a combination of probing methods such as ping and traceroute, and of heuristics to sample the IP address space in an intelligent way to find new nodes and links [17, 5].

The second class of methods relies on correlation in the packet loss process of a multicast session. Specifically, a packet loss in a multicast session is experienced by all the receivers downstream from the link where the loss occurred. Thus, by observing a large number of packets at these receivers, the structure of the multicast tree can be approximately inferred [3, 31]. This corresponds to finding a subgraph of the network topology.

The third class of methods focuses on inferring other static attributes of the network, such as the capacity of links through active probing [2], or the scheduling discipline [26]. On the Web, identifying and characterizing intermediaries (such as HTTP/1.0 and HTTP/1.1 proxies) is attempted via probing techniques.

Internal measurements There are several additional sources of information about network topology and configuration when one has administrative control over a network domain. Chief among these are the router configuration files, which provide a router's local view of the topology, including its neighbors, links to and from these neighbors and their capacities. From this, it is conceptually easy to completely determine the physical network topology of the domain [11]. Complications can arise because direct manipulation of router configuration files by operations personnel can lead to inconsistent configurations.

Inferring the topology at the AS level (a graph with currently approximately 10000 nodes) is impossible today. While it is easy to obtain a list of all the ASs, their connectivity depends on local public and private peering arrangements and the routing policies put in place by ISPs. Some heuristic methods to infer subgraphs of the full AS topology are described in [16, 8, 13].

2.2 State

Next, we compare inferring the state of the network, assuming that the underlying topology is known. Network state includes the operational state of links and routers, the routing and forwarding tables in effect, and other variables that do not directly depend on traffic (e.g., temperature of the CPU).

External measurements Essentially the same tools used for external topology discovery can be relied upon to discover the operational state of links and routers in a domain. The obvious drawbacks, as in any polling scheme, is that there is a potential delay between a state transition and the time it is discovered; this delay depends on the poll cycle. Also, the absence of a response to a ping packet can imply that the target router or interface is down, or that the path to that router/interface is down. Therefore, the results of multiple pings must be carefully combined to infer link and router state correctly.

To obtain a snapshot of the state of routing, traceroute has been successfully used [30]. This basically amounts to temporally sampling a small subset of routing table entries. Pathchar [19] is an extension of traceroute that is able to obtain rough estimates of additional path characteristics (loss and delay). Beyond this, it is virtually impossible to measure other state variables from the outside.

Internal measurements Observing the state of network elements is the realm of network management protocols such as SNMP [34]. SNMP enables a network management station to query remote state variables through an agent. The remote variables are standardized as a MIB (management information base) tree. In addition to this polling mode, SNMP allows the definition of events that alert the management system synchronously to state changes. In practice, SNMP tends to incur a relatively high overhead in routers, and its usefulness for fine-grained tracking of network state is limited.

Another method consists in intercepting link state advertisement messages exchanged by the intra-domain routing protocol (e.g., OSPF). This approach has the advantage of being authoritative, in the sense that the observed state is exactly the one that computation of routing tables is based upon [33].

2.3 Traffic

We next examine methods to measure the domain-wide traffic flow. This includes both the load (or demand) imposed on the network domain, the routes followed by the incoming traffic, and its loss and delay characteristics.

External measurements It is virtually impossible to estimate traffic as a whole through a large measurement domain through external probing of that domain. The only method that falls into this category are recent proposals for the inference of sink trees in distributed denial-of-service (DDoS) attacks, such as IP traceback [32]. In this method, routers randomly encode their address into the identification field of the IP header. With enough samples, a target site of a DDoS attack can reconstruct the sink tree of attack traffic through these encoded addresses.

Internal measurements There are several methods proposed in the literature that infer domain- wide traffic statistics from different types of measurement.

The first class of methods called *network tomography* relies only on measurements of link utilizations over time. The goal of these methods is to infer the traffic matrix, i.e., the traffic intensity between every ingress and every egress point [37, 4, 35].

The second class of methods relies on *aggregate flow measurements* at network ingress and/or egress points [10]. A flow is an artificial abstraction of a set of IP packets with identical source-destination addresses (or address prefixes) that are observed close together in time [6]. This method has some drawbacks in terms of overhead, implementation cost, and delay; nevertheless, careful post-processing can yield satisfactory estimates of the domain-wide traffic flow [10].

The third class of methods uses packet sampling. Packet sampling can either be used at all ingress points (in analogy to flow aggregation), relying on measured or simulated routing tables to infer the flow through the domain. Another method called *trajectory sampling* relies on pseudo-random sampling based on hash functions computed over packet content to directly observe these paths [7].

3 DNS and Web Measurements

The various problems we have described in the network layer are reflected largely in other layers as well. In general, having administrative access over all aspects in applications that cross the network layer is harder. As examples, we examine two application areas: Domain Name System (DNS) and the Web.

The most popular application on the Internet currently is the World Wide Web. In terms of traffic on the Internet, the Web is currently responsible for 75% of the packets on the Internet. The rate of growth of traffic between the millions of Web users and Web sites has grown steadily for a decade. Presently there is significant growth at the Intranet level as well. Traffic in peer to peer networks due to the popularity of Napster and Gnutella is growing but they are a much smaller part of the overall traffic and remain largely concentrated in college campuses.

Increase in user-perceived latency, redundant transfer of popular content across the network led to deployment of caches between the clients and the origin servers where resources reside or are generated. Once the usefulness of caching began to crest, offloading of content delivery became popular and led to the advent of content distribution networks (CDNs). Most CDNs use DNS-based redirection which has caused a significant increase in DNS traffic on the Internet.

We begin with some background information and then examine why inferring topology, state, and traffic is difficult at each of these areas.

3.1 DNS Measurements

Topology The topology of the DNS network consists of a collection of top-level domains (such as `.com`, `.edu`, `.it` etc.) that are just below the root of an hierarchy. These are then organized into separately administered zones (e.g., `att.com`). The individual zones are responsible only for registering the names and IP addresses of a set of authoritative DNS servers with the root servers.

Client requests for translation of names to IP addresses and vice-versa are typically sent by a resolver library that contacts a local DNS server. The local DNS server will check its cache for the request and if it does not have any pertinent information it will forward the request to a root server. The root server will return the names and addresses of the authoritative DNS server that can help answer the query. The queries may proceed iteratively with each query resulting in a pointer to the next server to be queried or recursively, whereby the queried server will do the necessary work and return the result. Positive caching (for hits) and negative caching (for failures) with a specific time to live value is routinely employed at the DNS servers. Most client sites have more than one local DNS server—one or two more serve as secondary servers for backup purposes. Figure 1 shows the various steps involved in resolving the address of the server component embedded in a URL

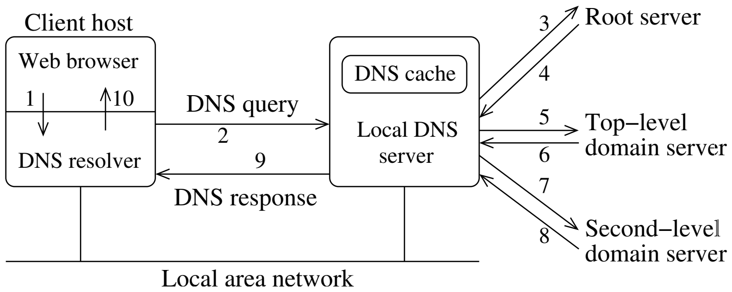

Fig. 1. DNS resolver and local DNS server

A host relying on DNS may be able to examine the static configuration file (e.g., `/etc/resolv.conf` on many UNIX systems) but more recent systems rely on DHCP (Dynamic Host Configuration Protocol) for more automated configurations. The extent of information available to a client is often limited to the names of local DNS servers. The set of authoritative DNS servers that may have to be contacted by the local DNS servers to resolve all the various addresses of interest is simply too large. The set of authoritative DNS servers change from time to time. The local DNS servers have to be kept operational at all times since in their absence applications will not be able to conduct any remote network activity.

Even when the set of DNS servers are within a single administrative domain, there is no simple mechanism to obtain a list of the configured collection since they are distributed across a large number of machines. An attempt to walk the DNS tree hierarchy by using tools like *dig* [1] starting at the zone from a server and examining the name server (NS) records would still miss servers that are not authoritative (caching-only servers) and the unofficial secondary servers used for fault tolerance purposes. Additionally, access control lists placed on zones will make this process even harder. The model of interaction with remote DNS servers is hop by hop with little or no knowledge available about anything beyond the first hop. There is no equivalent of traceroute at the DNS layer.

State Changes occur often at the DNS layer: new domains are registered, old ones change, cache information becomes stale, etc. The only way to learn about changes is upon a request and failure is often based on a timeout model. Consider the problem of a common mis-configuration known as *lame delegation*: a set of IP addresses have been registered as authoritative for a particular domain. Suppose one of them is incorrect: it is either not an authoritative server for that domain or does not even run a name server. Queries sent to this server will time out and be resent leading to additional unnecessary traffic. Note that a lame server may be contacted directly or as a result of redirection from another authoritative server for the zone.

It has been estimated that there may be up to 25% of all DNS zones with lame delegations. There are two problems in identifying and removing lame delegations. The lame delegations are rarely discovered since often the only indication of their presence is additional latency and a redundant server will eventually answer the query. Even if they are identified, fixing the problem requires interactions with administrators (who are hard to locate). Finally, there is no guarantee that the problem will indeed be fixed. Although the rate of change of information at the DNS layer is significantly less than at the network layer, issues of scale and distributed control make it harder to determine and correct state problems.

Traffic There are no known mechanisms to trace the DNS traffic as it percolates through the hierarchy. Although logs are maintained on several servers (primarily to detect scan attempts by hackers) they disclose at best partial information. The handing off of control to a backup server via DNS's zone transfer mechanism (a common occurrence) is not known to other servers since the mechanism is meant to be transparent to the end users. The recent significant increase in the DNS traffic due to overlay networks like Akamai (whereby URLs of content distributed resources are replaced with alternate CDN company specific ones whose address resolution can be controlled for load balancing purposes) became visible only because the rewritten URLs can be seen in the HTML text. Other CDNs do dynamic URL rewriting making such detection even harder. DNS servers of CDN companies often give out short TTLs to have more fine-grained control over the use of the mirror servers. With each short TTL expiration the CDN can balance the load on its network of servers. However, this results in additional DNS traffic with questionable performance improvements. Furthermore, obtaining a complete list of sites to which requests get redirected is hard since the CDN resolution is designed to give different answers at different times depending on the client's location.

Inside the network, one can obtain flow records via tools like *netflow* or by running packet tracing programs like *tcpdump*. In both cases, the task of extracting DNS traffic for purpose of identifying problems is pretty complex although there is currently some work in progress in this direction [24]. The problems at the internal level include obtaining a complete view of all DNS traffic entering and exiting the network. Even if the complete information is obtained at high cost, the local configurations are not known which results in at best a partial view of what is actually occurring.

3.2 Web Measurements

As mentioned earlier, Web traffic accounts for 75% of all packets on the Internet. A Web transfer requires interaction with a variety of protocols and often with multiple entities on the Internet. The suite of protocols include the Domain Name System (DNS) protocol, the transport layer protocol (often TCP) for transporting the requests and responses reliably between the Web client and the server, and HTTP (HyperText Transfer Protocol [12, 23]—the protocol underlying the World Wide Web and serves as the language of Web messages.

A single user click can result in the involvement of several entities [23, Chapter 15] including the following:

- A browser (and often a client side cache).
- Several Web intermediaries such as Web proxies or gateways. The proxy may be directly configured, be invisible to users (interception proxy), or be part of a large proxy farm.
- A surrogate server in front of the actual Web server deployed to balance the load at the Web site.
- The DNS server at the client/proxy side and additional redirected lookups due to rewritten URLs (due to content distribution overlay networks such as Akamai or Digital Island) and advertisement servers (who contribute images and text to the full container document).

The number of parties involved on an end to end basis in a single Web transaction can be more than a handful.

Topology The resources requested on the Web by clients like browsers (or quite often programs like spiders), can often be served from different locations either locally through surrogates on the server side or remotely at mirror sites. The set of entities on the Web include clients, proxies, gateways, servers, surrogates, mirrors. The choice of mirrors is dynamically decided. A user's request may traverse several of these entities. Often the user has only control over the next-hop proxy and thus identifying all the entities is hard. The recent advent of HTTP/1.1 version of the protocol [12, 23, 22] allows intermediaries to identify themselves through a new HTTP header (`Via`). But given the widespread prevalence of HTTP/1.0 proxies in the Internet for the foreseeable future, those entities may not participate in this enhancement. Furthermore, the presence of interception proxies (the ones that dip into the network layer to examine suspected HTTP traffic and possibly redirect them) exacerbates the difficulty of knowing all the entities that are involved in a transaction.

State A resource may be cached at a proxy or at a dynamic mirror site. For load balancing and fault tolerance purposes a set of resources may be available from multiple sites in case some sites are inaccessible at any given time. Learning about the state of a specific server is hard due to possible redirection at the HTTP or DNS level.

The widespread presence of caching proxies makes it much harder to determine the actual number of requests generated for a particular resource. Downloading a container document (such as an HTML file which includes links to one or more embedded resources, such as images or animations) requires contacting several servers due to the growing use of CDNs and the presence of advertisements (often located on remote machines). Since the CDNs often dynamically decide the mapping between strings and the actual machines that serve the distributed content, it is not possible to obtain stable latency metrics. The end to end measurements of interest include user perceived latency, load on the network, and load on the servers. However, inferences regarding load on a remote server is very hard to obtain. Simple hacks like examining TCP sequence numbers can be risky in the presence of redirections at the HTTP and DNS layer. Caches introduce the well known problem of staleness: a significant fraction of HTTP requests are validation queries to ensure that a cached resource is the same as the current instance on the origin server. There are risks to a cache assigning freshness time overriding the origin server's wishes.

Traffic The end to end traffic on the Web is both simply too large and too complex to estimate. Companies like Media Metrix and Keynote attempt to present sampled figures by examining traffic at a few interchanges and extrapolating from them. As discussed above, such studies miss the cached responses, failures, etc. Some studies have been carried out to perform end to end measurements [25] that examine improvements due to the new version of the protocol such as reduction in the number of TCP connections due to persistent connections feature, cache effectiveness, content delivery from multiple sites (CDNs, ad servers etc.), and latency reduction due to the ability of downloading partial responses. Yet, there is not a statistically reliable sampling technique for estimating end to end traffic, due to the complexity of the Web, the widespread prevalence of intermediaries, and implementations that are not compliant with the protocol specification, etc.

Even on an intra-net level, a Web server may not know what fraction of requests directed towards it reach it eventually due to the possibility of proxy cache farms in the path. The server would have to know about the configuration information of all clients in order to obtain a good estimate or indulge in cache busting [23].

Currently the best known traffic artifacts are logs maintained at the proxy and server level. The logs record several fields including the IP address of incoming request, time of request, the HTTP method, URL, protocol version, response code and content length. Even this relatively small subset of items logged has problems associated with how they are interpreted. The 'client' IP addresses recorded could be the last hop proxy and not the original client. A long response in transit may never reach the client who may have aborted the request; yet the server log might indicate that several thousands bytes of response were sent.

4 Conclusion

We have argued that network measurements are crucial both to drive fundamental discoveries and for the purpose of control and engineering. At the IP network layer, this need is fairly obvious, and instrumentation support at the network layer is reasonably mature as a result of pressure on vendors to include measurement support in their products. However, end-user performance depends as much on the performance of logical networks such as the DNS and the Web as on the network layer proper. This suggests that the granularity and scope of measurements available at these layers should match that at the network layer. We have illustrated in this paper that this is not the case through a direct comparison of the state of the art at the network layer with the DNS and the Web.

We have described network measurements for topology, state, and traffic. There is a range of methods for each category. Administrative access to a network domain is usually required to gain access to measurements of sufficient quality to perform traffic engineering. Nevertheless, a range of clever methods are also available to obtain useful snapshots of network topology, routing state, and traffic loads.

At the higher layers the problem of inference is more complicated since topology, state, and traffic have several additional entities and hidden artifacts (some of which are known, such as lame delegations at the DNS layer). Additionally, interesting questions such as user-perceived latency or end-to-end delay are inherently more complicated due to the involvement of multiple protocols and intermediaries. The problem is further compounded by implementations of Web components that are not fully compliant with the protocol specification [21]. In some cases application level protocols have attempted to mimic some of the useful ideas in the network layer. For example, HTTP/1.1 introduced the `Via` and `Max-Forwards` header to expose some of the topology information about intermediaries and to better target the requests.

We believe that the additional measurement support at the application layer could be inspired by tools and methods that have proved valuable at the network layer. For example, an equivalent of *traceroute* in DNS to track a request, *pathchar* in HTTP to derive performance properties of nodes (proxies) on the path to the server, or native support for request sampling, would be useful for troubleshooting, testing, and control. While the details of such a suite of higher-layer tools would certainly reflect the application area, we hope that the foregoing discussion has shown conceptual similarity between the network layer and other application areas to motivate the reuse of the expertise gained at the network layer.

References

1. Paul Albitz and Cricket Liu. *DNS and BIND*. O'Reilly, third edition, September 1998. ISBN 1565925122.
2. J-C. Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proc. ACM SIGCOMM '93*, San Fransisco, CA, September 93.
3. R. Cáceres, N.G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley. Loss-based Inference of Multicast Network Topology. In *Proc. 1999 IEEE Conference on Decision and Control*, Phoenix, AZ, December 1999.
4. J. Cao, D. Davis, S. Vander Wiel, and B. Yu. Time-Varying Network Tomography. *Journal of the American Statistical Association*, December 2000.
5. Bill Cheswick, Hal Burch, and Steve Branigan. Mapping and Visualizing the Internet. In *Usenix 2000*, San Diego, CA, 2000.
6. Cisco Corp. Netflow Services and Applications (white paper). August 1999.
7. N. C. Duffield and M. Grossglauser. Trajectory Sampling for Direct Traffic Observation. *IEEE/ACM Transactions on Networking*, June 2001.
8. Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. In *Proc. ACM SIGCOMM '99*, pages 251–262, August/September 1999.
9. A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford. NetScope: Traffic engineering for IP networks. *IEEE Network Magazine*, March 2000.
10. A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. In *Proc. ACM SIGCOMM*, Stockholm, Sweden, August 2000.
11. Anja Feldmann and Jennifer Rexford. IP network configuration for intradomain traffic engineering. *to appear in IEEE Network Magazine*, 2001.
12. R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol — HTTP/1.1. RFC 2616, IETF, June 1999. Draft Standard of HTTP/1.1. `http://www.rfc-editor.org/rfc/rfc2616.txt`.
13. Lixin Gao. On inferring autonomous system relationships in the Internet. In *Proc. Global Internet 2000*, November 2000.
14. R. J. Gibbens, F. P. Kelly, and P. B. Key. A Decision-theoretic Approach to Call Admission Control in ATM Networks. *IEEE Journal on Selected Areas of Communications*, pages 1101–1114, August 1995.
15. Gnutella. `http://gnutella.wego.com`.
16. Ramesh Govindan and Anoop Reddy. An analysis of Internet inter-domain topology and route stability. In *Proc. IEEE INFOCOM '97*, April 1997.
17. Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for Internet Map Discovery. In *Proc. of IEEE Infocom 2000*, Tel Aviv, Israel, April 2000.
18. M. Grossglauser and D. N. C. Tse. A Framework for Robust Measurement-Based Admission Control. *IEEE/ACM Transactions on Networking*, 7(3), June 1999.
19. Van Jacobson. Pathchar. `ftp://ftp.ee.lbl.gov/pathchar`.
20. S. Jamin, P. B. Danzig, S. Shenker, and L. Zhang. A Measurement-Based Admission Control Algorithm for Integrated Services Packet Networks. *IEEE/ACM Transactions on Networking*, 5(1), February 1997.
21. Balachander Krishnamurthy and Martin Arlitt. PRO-COW: Protocol Compliance on the Web—A Longitudinal Study. In *Proc. USENIX Symposium on Internet*

Technologies and Systems, March 2001.
`http://www.research.att.com/~bala/papers/usits01.ps.gz`.
22. Balachander Krishnamurthy, Jeffrey C. Mogul, and David M. Kristol. Key Differences between HTTP/1.0 and HTTP/1.1. In *Proc. Eighth International World Wide Web Conference*, May 1999.
`http://www.research.att.com/~bala/papers/h0vh1.html`.
23. Balachander Krishnamurthy and Jennifer Rexford. *Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement*. Addison-Wesley, May 2001. ISBN 0-201-710889-0.
24. Balachander Krishnamurthy, Oliver Spatscheck, Chuck Cranor, Emden Gansner, and Carsten Lund. Characterizing DNS traffic. Document in preparation.
25. Balachander Krishnamurthy and Craig E. Wills. Analyzing Factors that influence end-to-end Web performance. In *Proc. World Wide Web Conference 2000*, pages 17–32, May 2000.
`http://www.research.att.com/~bala/papers/www9.html`.
26. A. Kuzmanovic and E. Knightly. Measuring Service in Multi-Class Networks. In *Proc. of IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.
27. Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM Trans. on Networking*, 2(1):1–15, February 1994.
28. Andy Myers, Peter Dinda, and Hui Zhang. Performance Characteristics of Mirror Servers on the Internet. In *IEEE INFOCOM '99*, New York City, March 1999.
29. Napster.
`http://www.napster.com`.
30. V. Paxson. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, October 1997.
31. S. Ratnasamy and S. McCanne. Inference of Multicast Routing Trees and Bottleneck Bandwidths using End-to-End Measurements. In *IEEE INFOCOM '99*, New York City, April 1999.
32. Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical Network Support for IP Traceback. In *ACM SIGCOMM 2000*, September 2000.
33. Aman Shaikh, Mukul Goyal, Albert Greenberg, Raju Rajan, and KK Ramakrishnan. An OSPF Topology Server: Design and Evaluation. *submitted for publication*, 2001.
34. William Stallings. *SNMP, SNMP v2, SNMP v3, and RMON 1 and 2 (Third Edition)*. Addison-Wesley, Reading, Mass., 1999.
35. C. Tebaldi and M. West. Bayesian Inference on Network Traffic. *Journal of the American Statistical Association*, June 1998.
36. V. Paxson and S. Floyd. Why We Don't Know How To Simulate The Internet. In *Proceedings of the 1997 Winter Simulation Conference*, December 1997.
37. Y. Vardi. Network Tomography. *Journal of the American Statistical Association*, March 1996.