HTTP Simulator Validation Using Real Measurements: A Case Study

Brian D. Davison[*]
Department of Computer Science
Rutgers, The State University of New Jersey
davison@cs.rutgers.edu

Abstract

Simulation is a common technique used by Web researchers and provides many benefits. Verification and validation of a simulator are essential if the results of those simulations are to be believed. Unfortunately, even limited validation of simulators has been uncommon in the Web caching community. In contrast, this paper argues for the validity of a new network and caching simulator by extensively comparing simulated results to both small– and large-scale real-world HTTP traffic. In addition, we describe some of the preparation needed to use a large, well-known trace of Web usage.

1 Introduction

Simulation is a common approach for algorithmic research, and is widespread in Web caching research (e.g. see [4]). In general, simulation provides many benefits, including the ability to:

- test scenarios that are dangerous or have not yet occurred in the real world.
- predict performance to aid technology design.
- predict expected behavior of new protocols and designs without implementation costs or disruption of existing systems.
- slow or speed time for effective analysis.
- quickly survey a range of potential variations.

[*]New address: Department of Computer Science and Engineering, Packard Lab, Lehigh University, 19 Memorial Drive West, Bethlehem, PA 18015.

A simulator is only useful when it produces *believable* results. Believability, however, is a subjective quality, and typically only comes after the simulator has been validated by exhaustive testing and widespread use. Heidemann *et al* [14] provide one definition of validation as the "process to evaluate how accurately a model reflects the real-world phenomenon that it purports to represent." In this paper we aim to validate portions of network models embodied in a new simulator called NCS (Network and Cache Simulator) [6].

Verification and validation [23] are essential if the results of those simulations are to be believed. This is particularly important when the simulated experiments are not easily replicated for confirmation of simulated results (perhaps because of extensive development effort required or proprietary log data). Lamentably, even limited validation of simulators has been uncommon in the Web caching community.

In this paper we will simulate HTTP network performance at both a small scale (individual transactions, carefully measured) and at a large scale (tens of thousands of clients and servers from a log real-world usage). While NCS supports caching, in this work we focus strictly on the networking aspects which can be validated by usage logs that do not include caching effects that need to be modeled. It is our intent to demonstrate the validity of the HTTP retrieval latencies as estimated for the client in our HTTP simulator by comparison to real-world data. However, within this effort, we effectively provide a case-study of two processes with pedagogical merit independent of the simulator used. One is the preparation and analysis of a large, well-known HTTP usage trace. The other is the process of simulator validation by comparison to real-world data.

The rest of the paper is organized as follows: Section 2 provides relevant background such as the simulator being used and the process by which we will test the simulator for validity. The strongest validation we can provide is to be able to replicate real-world results. Thus, in section 3 we replicate under simulation the network effects of some small-scale real-world experiments. Such a small scale comparison is useful and necessary for verification, but the overall goal is to be able to generate results on relatively large experiments. Therefore, in section 4 we describe a large dataset of real-world traffic, and use it in section 5 for comparisons with simulated results. We wrap up with future validation work and a summary.

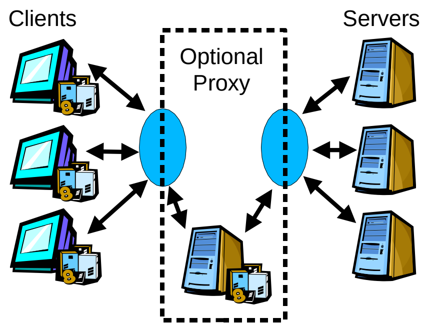


Figure 1. NCS topology.

2 Background

2.1 NCS: Network and Cache Simulator

NCS is an HTTP trace-driven discrete event simulator of network and caching activity. It is highly parameterized for maximal compatibility with previous caching and network simulations. In granularity, it resides between the high-level caching-only simulators typical of much of Web caching research, and the detailed simulators of networking protocols and traffic. In an effort to capture estimates of user-perceived latency, it simulates simplistic caching and prefetching functionality at various locations in a network comprising of client browsers, an optional intermediate proxy, and Web servers (shown in Figure 1). NCS dynamically creates clients and servers from templates as they are found in the trace. Caching at the proxy and clients is optional. Additionally, it simulates many aspects of TCP traffic among these entities on a somewhat idealized network.

The development goals for NCS included:

- Estimate client-side latencies and bandwidth usages by 1) capturing intrinsic network effects (e.g. new connection costs, TCP slow start); 2) modeling real topologies (including browsers, proxies, and servers, with potentially multiple connections and/or persistent connections); and 3) capturing some real-world network effects (including distributions of latencies and bandwidths).

- Provide credibility – be able to compare simulation results to real-world numbers.

- Incorporate optional prefetching techniques for testing and evaluation.

While most Web caching simulators measure hit rates and bandwidth used, few consider the detail needed to believably estimate retrieval latency. In contrast, NCS is specifically designed to estimate the latency experienced by the user, and so includes network simulation in addition to a caching implementation.

Although inspired by the caching simulators described and used in [10, 9, 7], the development of NCS has proceeded independently. While this has necessitated significant redevelopment, it has had the side-benefit of being an educational process for the author. An alternative might have been to use the network simulator *ns* [25] and extend it appropriately for prefetching. However, *ns* is also known to have a steep learning curve [3], and would likely require significantly more computational resources because of the more detailed modeling it performs. By using a less-detailed network model, NCS is able to estimate performance for traces using tens of thousands of hosts in much faster than real-time. Additionally, NCS uses trace-based workloads with the associated characteristics of actual data, instead of artificially generated data. In summary, we wanted more detail (and response-time accuracy) than typical caching-only simulators, but faster simulation times for large experiments than the fine-grained network-oriented simulator *ns*, and to be able to incorporate code to optionally estimate the effects of prefetching (which we do not explore here).

A more comprehensive description of NCS along with sample results is available elsewhere [6].

2.2 Related Work

Many researchers use simulation to estimate Web caching performance. Often they measure object and byte hit rates and ignore response time latencies. Response time improvement, however, is a common justification for the use of Web caching, and is arguably the initial *raison d'être* for content delivery networks. As mentioned above, however, there are other simulators that estimate response times. We describe three of the most prominent ones here.

PROXIM [10] is a caching and network effects simulator developed by researchers at AT&T Labs. Like NCS, PROXIM accounts for TCP slow start and does not consider packet losses and their resulting effects. However, PROXIM does not including prefetching, and their paper only provides for a simple level of validation — showing similar curves for the simulated latency distribution as the original latency distribution, and comparing median total latencies. The traces used by Feldmann *et al* were captured by a snooping proxy, much like those the trace we will describe below in section 4, and have likely been cleaned and adjusted similarly (although this isn't described explicitly). One unusual aspect of these experiments is that the traces provide extraordinary timing details (including timestamps of TCP as well as HTTP events), allowing PROXIM to use RTT estimates from measurements of SYN to SYN-ACK and HTTP request and response timestamps on a per-connection basis.

Like NCS, the simulator described in Fan *et al* [9] uses the timing information in a portion (12 days) of the UCB trace (which we describe in section 4) to estimate latency seen by each modem client. However, it also uses a much simpler model of latency that

- estimates client-seen latency to be the sum of 1) time between seeing the request and the first byte of the response, and 2) the time to transfer the response over the modem link.
- apparently ignores connection setup costs.
- apparently ignores TCP slow start effects.
- groups overlapping responses together.

However, Fan *et al* claim that by grouping overlapping responses together they are able to measure the time spent by the user waiting for the whole document. No description of verification or validation of the simulator is provided.

The *ns* simulator [25] is likely the best-known networking simulator, but is not typically used for caching performance measurements, possibly because of slow simulation speeds. It uses detailed models of networking protocols to calculate performance metrics (see Breslau *et al* [3] for an overview). This simulator has been validated by widespread user acceptance and extensive verification tests [11].

2.3 The Validation Process

Typically it is impossible for a simulator to exactly mirror real-world experience. NCS is no exception. NCS ordinarily uses fixed parameters for many aspects of client, network, and server performance. In the real world, these values would be a function of changing world conditions. Different servers would have different connectivity characteristics (including network latency and bandwidth) and service loads at different times. Dial-up clients will likewise differ in bandwidths and latencies depending on the hardware, software, and phone lines used. NCS only models the traffic captured in the input trace, ignoring unlogged traffic (e.g. non-HTTP traffic like HTTPS, streaming media, FTP, or traffic destined for non-logged clients or servers).

Given that the simulation will not generally *replicate* real-world experiences, what can be done? Instead, we can use the simulator to repeat simple real-world experiments, and thus validate the simulator by comparing it to both real-world results and the calculated results of others (which we do in section 3). In particular, we can ask whether the same effects are visible, and most importantly, verify that the simulator works as expected (in general, similarly to other results with some deviation as a result of the simplifications made).

3 Small-Scale Real-World Networking Tests

Fortunately, some networking researchers have taken the time to validate their networking models by comparing them to real-world results (e.g. [15]). By publishing such work in detail, others are not only able to validate new simulators, but also to compare fidelity of various theoretical models to those results.

In this section we do exactly this. We use the real-world measurements reported in published papers [15, 20] and attempt to reproduce the real-world experiments under simulation. Heidemann *et al* [15] use two sets of experiments for validation of their model of various implementations of HTTP over TCP. The first measures the overall retrieval time of a small cluster of resources.[1] This workload is tested in two environments (Ethernet and a high-speed Internet connection) using two forms of the HTTP protocol (HTTP/1.0 and HTTP/1.0 with persistent connections). The actual measurements were gathered and used by Heidemann *et al*.

The second set of experiments measures the overall retrieval time of a 42KB Web page with 42 embedded images totaling 125KB. This workload is tested in multiple environments (including Ethernet, high-speed Internet, and modem), but Heidemann *et al* only considered validating one protocol (HTTP/1.1-style with pipelining). The measurements (along with others) were collected and published by a different group [20].

We attempted to configure our simulator to be as realistic as possible. Beyond the basic modeling of HTTP over TCP/IP that Heidemann *et al* describe [15], we incorporated additional costs. We estimated the cost of a establishing a new connection based on transmission of a 40B packet (i.e. just TCP/IP headers) round-trip and a CPU time cost of .1ms. We model the appropriate number of parallel connections. We also modeled a fixed reply header with an estimated size of 200B. Finally, we reduced bandwidth estimates in the modem cases to account for TCP/IP and PPP error correction framing bytes.

In Table 1 we show the results of our simulator and compare it to the measurements made by Heidemann *et al*. As can be seen from the ratio column, the simulated results are quite close to the real measurements. They also improve upon the adjusted modeled values predicted in Table 7 of [15] in three cases, and equal the fourth (reducing the average error from 11.5% to 8.2%). From these results it appears that we may have underestimated the cost of establishing a new connection.[2]

[1] These resources were a single 6651B page with embedded 3883B and 1866B images, corresponding to the Yahoo home page on May 1, 1996

[2] A more appropriate estimate of the cost of a new connection may be on the order of multiple ms to account for slower machines and relatively unoptimized code back in 1996. [10] reports the mean total connection setup time in their modem data as 1.3s, which suggests the presence of significant server delays.

Environment		NCS	[15]	ratio
protocol	**network**	**simulated**	**measured**	**m:s**
HTTP/1.0	Ethernet	29.3ms	36.8ms (10ms +/12.0ms)	1.26
HTTP/1.0+KeepAlive	Ethernet	26.8ms	26.6 (8.8ms +/-1.7)	0.99
HTTP/1.0	Fast-Internet	1726ms	1716 (101ms +/-20.1)	0.99
HTTP/1.0+KeepAlive	Fast-Internet	1166ms	1103 (48ms +/-9.5)	0.95

Table 1. Validation of NCS on small cluster workload. Examines HTTP/1.0 (serial retrievals on separate connections) and HTTP/1.0+KeepAlive (serial retrievals on a single connection). Measured shows the mean of 100 trials with standard deviation and 95% confidence intervals in parentheses from [15]. m:s ratio is the ratio of measured time vs. simulated time.

Environment			NCS	[20]	ratio	ratio
protocol	**network**	**client/server**	**simulated**	**measured**	**m:s**	**m:a**
HTTP/1.1+Pipelining	Ethernet	libwww/Apache	163.4ms	490ms	3.00	1.47
HTTP/1.1+Pipelining	Fast-Internet	libwww/Apache	1622ms	2230ms	1.37	1.25
HTTP/1.1+Pipelining	Modem	libwww/Apache	53073ms	53400ms	1.01	1.00

Table 2. Validation of NCS on large cluster workload. Examines performance for HTTP/1.1+Pipelining on a single connection. Measured shows the mean of 5 trials from [20]. m:a ratio is the ratio of measured time to adjusted simulated time.

Table 2 compares the results of NCS to the measurements from [20]. In this test of pipelining, the simulator performs poorly for Ethernet, but respectably on the Fast-Internet and Modem cases. We suspect this is the result of a number of factors: 1) we model CPU time costs as a delay (which may occur in parallel with transmission of other data) rather than a use of resources (on the server side) during which no transmission likely takes place; 2) we do not know precisely the sizes nor ordering of the images, and instead use equal-sized values; 3) we do not know where within the first HTML retrieval that the embedded image references are placed, and so assume that the client can attempt to retrieve all objects at start; and 4) we do not model the HTTP pipeline buffering effects described in [20]. These effects are more pronounced for the Ethernet case because the scale of measurements is much smaller. The column marked ratio m:a shows what the ratio would be if we were to account for factors 1, 3 (assuming a delay of two packets), and 4 (adding a single 50ms delay as described by Nielsen *et al*). The adjusted values show improvement compared to the adjusted modeled values predicted in Table 8 of [15] in all cases (reducing the average error from 39.3% to 24.3%, although [15] was validating a slightly different dataset from an earlier version of the paper by Nielsen *et al*).

Since [20] include many more measurements than those used by Heidemann *et al*, we provide additional comparisons in Table 3. Again, the simulated values are not far from the measured values, except for the case of Ethernet (with likely similar reasons as those described earlier).

Over these datasets, NCS performs similarly to the model used by Heidemann *et al* and on average is slightly closer to the actual measurements reported. Thus we can conclude that at least in the small, NCS is likely to provide a reasonable estimate of real-world delays attributable to TCP.

4 The UCB Home-IP Usage Trace

In this section we describe the UC Berkeley Home IP HTTP Traces [12] and some of the effort needed to prepare the logs for our use. This dataset was not selected arbitrarily, but was chosen specifically because of its non-trivial length, recorded timing characteristics, and because it is well-known by researchers. A longer version of this trace has been analyzed in depth by Gribble and Brewer [13], but the version described here has also been used in numerous published papers (e.g. [8, 16, 9, 2, 24, 1, 18, 19, 21, 22]).

4.1 Background

The UC Berkeley Home IP HTTP Traces [12] are a record of Web traffic collected by Steve Gribble as a graduate student in November 1996. Gribble used a snooping proxy to record traffic generated by the UC Berkeley Home IP dialup and wireless users (2.4Kbps, 14.4Kbps, and 28.8Kbps land-line modems, and 20-30Kbps bandwidth for the wireless modems). This is a large trace, covering 8,377 unique clients over 18 days with over nine million requests. His system captured all HTTP packets and recorded, among other items, the time for each request, the first byte of re-

Environment			NCS	[20]	ratio
protocol	**network**	**client/server**	**simulated**	**measured**	**m:s**
HTTP/1.0	Ethernet	libwww/Apache	193ms	720ms	3.73
HTTP/1.1	Ethernet	libwww/Apache	358ms	810ms	2.27
HTTP/1.0	Fast-Internet	libwww/Apache	5402ms	4090ms	0.76
HTTP/1.1	Fast-Internet	libwww/Apache	5628ms	6140ms	1.09
HTTP/1.1	Modem	libwww/Apache	62672ms	65600ms	1.05
HTTP/1.0+KeepAlive	Modem	Netscape 4/Apache	53523ms	58700ms	1.10
HTTP/1.0+KeepAlive	Modem	Internet Explorer 4/Apache	53823ms	60600ms	1.13

Table 3. Validation of NCS on large cluster workload. Examines performance for HTTP/1.0 (with up to 4 parallel connections), HTTP/1.1 (with a single persistent connection), and HTTP/1.0+KeepAlive (with up to either 4 or 6 parallel connections for Netscape or MSIE respectively). Measured shows the mean of 5 trials, except for the last two rows where it is the mean of 3 trials, from [20].

sponse, and the last byte of response. These timestamps provide a sample of real-world response times that can be used to validate our simulator.

4.2 Trace Preparation

Most researchers have found that web traces need to be checked and often cleaned before using them in a simulator or for evaluation [17, 5]. The UCB Home-IP Trace is no exception.

This trace does not record the HTTP response code associated with each object. Thus, we are unable to distinguish between valid responses (e.g. code 200), error responses (e.g. 404), and file not modified responses (304). For the purpose of simulation, we will assume all responses contain a 200 response code.

While attractive for timing simulations, this trace also includes some anomalies. On example of this is impossibly high client bandwidths: for certain responses, the combination of size of reply and timings of request and end of response suggest bandwidths that meet or exceed LAN capabilities, and which certainly do not reflect dialup or wireless clients of 1996.[3] More importantly, the trace does not directly reflect client-perceived response times which is the timing estimate provided by NCS. Since the trace is captured by a snooping proxy on an Ethernet on the way to the university's Internet connection, two modem (dialup or wireless) round trips are not captured (first round trip initiates the connection, second corresponds to the time to make a request and receive the last response), nor is the modem transmission time of the initial packets to open a connection. There is also the transmission cost of the request packet and final data packet (or more, as will be discussed in the next section). Therefore, in order to compare the original trace timings to our simulated client response times, we will add a factor of up to 667ms, reflecting typical timings for a 28.8 modem (which underestimates the higher latencies of wireless modems as well as slower dialup modems).[4] Prior to the adjustment, the trace had a mean response time of 17.4s and a median of 2.9s; afterwards it had a mean and median of 18.0s and 3.6s, respectively.

[3]Therefore, we drop more than half a million anomalous entries from the trace (primarily those with missing timestamp fields, corresponding to missed packets or cancelled requests, but also 13 entries with server-to-proxy bandwidth over 80Mbps, and 7237 entries with size $>$ 100MB).

4.3 Analysis

We analyzed statistics and distributions of the UCB Hope-IP Trace to better understand the activity represented, and to look for anomalies. First we consider the response size distribution. We found that 25% of all responses were 430 bytes or smaller, and that the median response size was approximately 2310 bytes. This compares to a mean response size of 6.7KB.

An obvious measure of user-oriented performance for a Web system is the response time. In examining this trace, we found that the first quartile of the response time distribution is at 1.0s, and the median is at 3.6s. The mean response time is much higher at 18.0s.

In order to better understand the distributions involved, we also examined the relationship between the size of a file transfered, and the response time for that file. In Figure 2, the actual response time vs. size of file is plotted for the first 100,000 requests in the UCB trace. We also plot the idealized performance of a 28.8 modem. Note the large number

[4]The value 667ms can be calculated as follows. We assume a modem round-trip time of 200ms, and effective bandwidth of 3420 bytes/sec (to account, in part, for the overhead of TCP/IP header bytes and PPP framing bytes). The transmission time to send and receive the initial packet to set up a new connection is 22.2ms (40 bytes sent and received at 3600 bytes/sec), the time to transmit the request is 95ms (325 bytes at 3420 bytes/sec), and the time to receive the last packet is estimated to be 149.7ms (512 bytes at 3420 bytes/sec). The total delay is then the sum of two round-trip times, the transmit time to send and receive initial packet to set up a new connection, the time to transmit a request, and the time to send a final response packet.

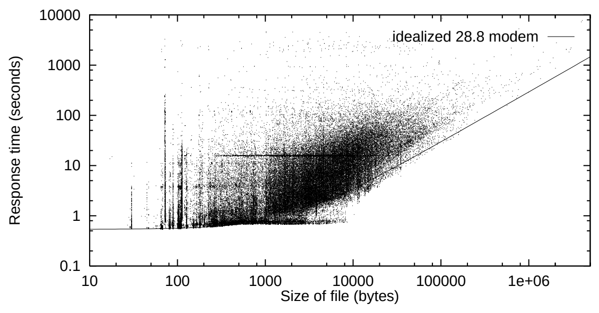


Figure 2. Scatter plot of file size vs. adjusted response time of actual responses from the UCB trace.

of points along the lower-right edge that correspond to transmission bandwidths that are higher than expected (below the 28.8 modem line). For example, there are many points at 8000 bytes with a response time of less than 1s. Assuming 1s, this corresponds to 64000 bits per second, which is clearly above the bandwidth limits for dialup and wireless technologies of 1996. We have two conjectures that provide plausible explanations. One is that these points (contributing to the fuzziness along the lower right edge) is an artifact of packet buffering at the terminal server interface (to which the modems were attached). The second possibility is that we are seeing the effect of data compression performed by the modems. Note that if we were to plot the original dataset (without filtering obvious anomalies), we would see even more extremal points to the lower right.

5 Large-Scale Real-World Networking Tests

By making reasonable estimates of network and host parameters (shown in Table 4), we can replay this trace within the simulator, and compare the resulting response times.

Parameter	Value
packet size	512 bytes
request size	325 bytes
client network latency (one-way)	100 ms
client bandwidth	3420 Bps
server network latency (one-way)	5 ms
server bandwidth	16384 Bps
server per request overhead	30 ms
cost of new connection	22 ms
max. conns. from client to host	4
latency distribution for hosts	Pareto
latency distribution for connections	Pareto

Table 4. Some of the simulation parameters used for replication of UCB workload.

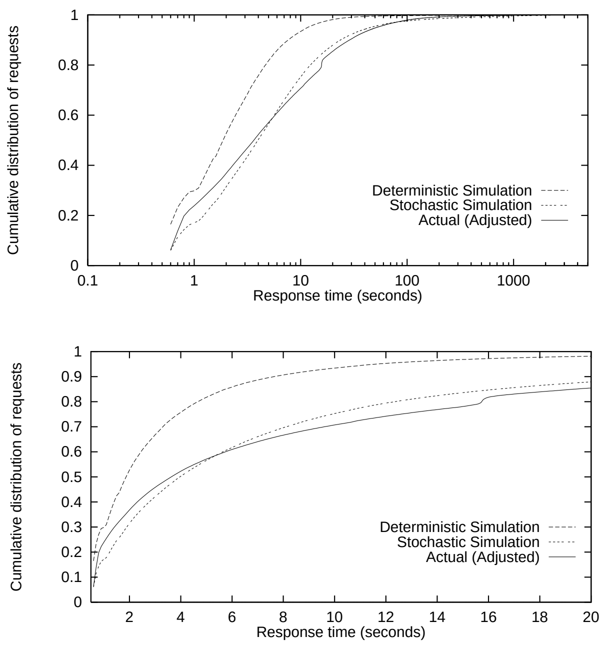


Figure 3. Comparison of the CDF of response times from the UCB Home-IP request trace (a, above) along with a magnification of first 20s (b, below).

Figures 3(a) and (b) show the cumulative distribution of response times for the actual trace and the results of two simulations (a deterministic trace and a stochastic trace that used heavy-tailed distributions of latencies on a per-host and per-connection basis). Likewise, we show results from the same experiments in Figure 4 which compare the distributions of effective throughput. The closeness of all of these graphs helps to validate the simulator. Unfortunately, the results will not be identical for a number of reasons:

- The simulator does not estimate the same thing that the snooping proxy measured. The 667ms adjustment mentioned above in section 4.2 is only a gross modification, and is not likely to adequately address the timings missing from the variety of client access devices, nor account for the differences in timings that result from significant buffering performed by the terminal servers nor modem compression.
- NCS does not model all aspects of the Internet nor does it model all traffic on relevant network links.
- The simulator does not match the actual host and network characteristics. Under the deterministic version of the simulator, each node is given the same network and host characteristics, which does not model the variations in connectivity (or responsivity of Web hosts).

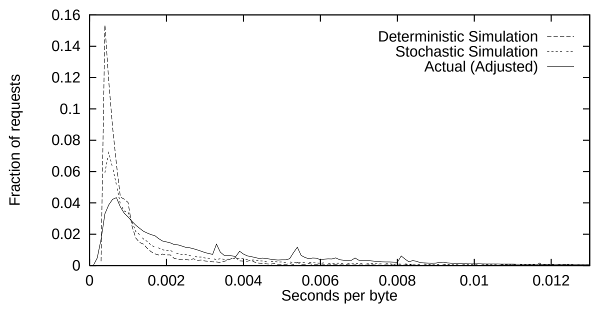


Figure 4. Distribution of simulated and actual effective throughput from the UCB trace.

Under the stochastic version, while these characteristics are determined in part by sampling from particular distributions, the activity of a client may actually depend on the connectivity which is being set independently.

- The parameters for the simulations have not been exhaustively explored, suggesting that there may be parameter values that would produce results even closer to the trace results.

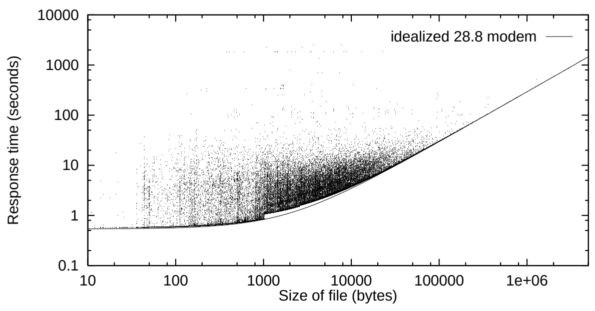
As in Figure 2, we have also displayed file size vs. response time from a simulated run, using static values for bandwidths and latencies, shown in Figure 5(a). In this scatter-plot, a well-defined lower right edge is visible, corresponding to the best response-times possible with the given bandwidths and latencies. The notch visible at approximately 1024 bytes corresponds to the point at which three packets are needed rather than two to transmit the file.

The obvious differences between Figures 2 and 5(a) suggest that a single static specification of bandwidths and latencies for clients and servers is insufficient to generate a sufficiently wide distribution of response times. Figure 5(b), on the other hand, does show more variance in the distribution of response times, qualitatively closer to the actual distributions shown in Figure 2, albeit with a defined lower right edge.

Thus Figure 5(b) may be considered "closer" to real-world performance than Figure 5(a), and as a result, demonstrate the trade-off of fidelity vs. complexity. While using heavy-tailed stochastic models may provide increased fidelity (in some sense), it increases the complexity of experiments that use the simulator (because of the need to perform repeated experiments to eliminate the stochasticity as the cause of the effect under consideration). The summary statistics are also closer, as shown in Table 5. In any case, we maintain that NCS is capable of generating aggregate response time results that are comparable to real-world results, even in large-scale experiments.

Trace	Median	Mean
Adjusted actual	3.6s	18.0s
Deterministic sim.	1.9s	5.0s
Stochastic simul.	4.0s	25.7s

Table 5. Basic summary statistics of the response times provided by the actual and simulated trace results.



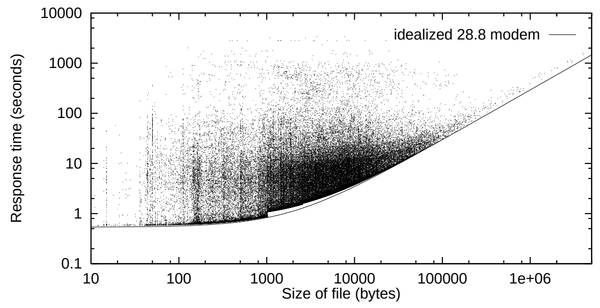


Figure 5. Scatter plot of file size vs. simulated response time for UCB trace. Above (a) shows deterministic configuration. Below (b) incorporates stochasticity.

6 Conclusion

While we expect that true believability will come with hands-on experience, we have used this paper to argue for the validity and believability of NCS for simulating retrieval latencies. In the process, we have provided a case study for network simulator validation by comparison to real-world data and discussed some of the preparation required for using the UCB Home-IP HTTP Trace.

Additional testing is always possible, but at some point becomes infeasible or unnecessary [14]. Given sufficient time and effort, there are two directions to which we see value. One is the generation of a simplified theoretical model with an analytical solution that can be compared to the simulation results. The second is the explicit comparison to existing simulations using identical data sets and assistance from the authors of validated simulators. These are desirable validations that we hope to explore in the future.

Acknowledgments

Thanks are due to Chandrasekar Krishnan, Chris Mesterharm and Ricardo Bianchini for valuable suggestions. This work was supported in part by the National Science Foundation under grant NSF ANI 9903052.

References

[1] B. R. Badrinath and P. Sudame. Gathercast: The design and implementation of a programmable aggregation mechanism for the Internet. In *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCN)*, Oct. 2000.

[2] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of IEEE INFOCOM*, New York, Mar. 1999.

[3] L. Breslau, D. Estron, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[4] B. D. Davison. A Survey of Proxy Cache Evaluation Techniques. In *Proceedings of the Fourth International Web Caching Workshop (WCW99)*, pages 67–77, San Diego, CA, Mar. 1999.

[5] B. D. Davison. Web traffic logs: An imperfect resource for evaluation. In *Proceedings of the Ninth Annual Conference of the Internet Society (INET'99)*, June 1999.

[6] B. D. Davison. NCS: Network and Cache Simulator – An Introduction. Technical Report DCS-TR-444, Department of Computer Science, Rutgers University, 2001. In preparation.

[7] B. D. Davison and V. Liberatore. Pushing Politely: Improving Web Responsiveness One Packet at a Time (Extended Abstract). *Performance Evaluation Review*, 28(2):43–49, Sept. 2000. Presented at the Performance and Architecture of Web Servers (PAWS) Workshop, June 2000.

[8] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *Computer Communication Review*, 28(4), Oct. 1998. Proceedings of ACM SIGCOMM.

[9] L. Fan, Q. Jacobson, P. Cao, and W. Lin. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '99)*, Atlanta, GA, May 1999.

[10] A. Feldmann, R. Cáceres, F. Douglis, G. Glass, and M. Rabinovich. Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments. In *Proceedings of IEEE INFOCOM*, pages 106–116, New York, Mar. 1999.

[11] S. Floyd. Validation Experiences with the NS Simulator. In *Proceedings of the DARPA/NIST Network Simulation Validation Workshop*, Fairfax, VA, May 1999.

[12] S. D. Gribble. UC Berkely home IP HTTP traces. Online: `http://www.acm.org/sigcomm/ITA/`, July 1997.

[13] S. D. Gribble and E. A. Brewer. System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS '97)*, Dec. 1997.

[14] J. Heidemann, K. Mills, and S. Kumar. Expanding confidence in network simulation. Research Report 00-522, USC/Information Sciences Institute, April 2000.

[15] J. Heidemann, K. Obraczka, and J. Touch. Modeling the performance of HTTP over several transport protocols. *IEEE/ACM Transactions on Networking*, 5(5):616–630, Oct. 1997.

[16] J. Judge, H. Beadle, and J. Chicharo. Sampling HTTP Response Packets for Prediction of Web Traffic Volume Statistics. In *Proceedings of IEEE Globecom*, Nov. 1998.

[17] B. Krishnamurthy and J. Rexford. Software Issues in Characterizing Web Server Logs. In *World Wide Web Consortium Workshop on Web Characterization*, Cambridge, MA, Nov. 1998. Position paper.

[18] J. Lilley, J. Yang, H. Balakrishnan, and S. Seshan. A unified header compression framework for low-bandwidth links. In *Proceedings of MobiCom: Sixth Annual International Conference on Mobile Computing and Networking*, Boston, Aug. 2000.

[19] J.-M. Menaud, V. Issarny, and M. Banâtre. Improving Effectiveness of Web Caching. In *Recent Advances in Distributed Systems*, volume 1752 of *LNCS*. Springer Verlag, 2000.

[20] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, and C. Lilley. Network performance effects of HTTP/1.1, CSS1, and PNG. *Computer Communications Review*, 27(4), Oct. 1997. Proceedings of SIGCOMM '97. Also available as W3C NOTE-pipelining-970624.

[21] S. Paul and Z. Fei. Distributed caching with centralized control. In *Proceedings of the Fifth International Web Caching and Content Delivery Workshop (WCW'00)*, Lisbon, Portugal, May 2000.

[22] D. Rosu, A. Iyengar, and D. Dias. Hint-based Acceleration of Web Proxy Caches. In *Proceedings of the 19th IEEE International Performance, Computing, and Communications Conference (IPCCC 2000)*, Phoenix, AZ, Feb. 2000.

[23] R. G. Sargent. Verification and validation of simulation models. In D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, editors, *Proceedings of the Winter Simulation Conference*, 1998.

[24] R. Tewari, M. Hahlin, H. M. Vin, and J. S. Kay. Design Considerations for Distributed Caching on the Internet. In *Proceedings of the 19th International Conference on Distributed Computing Systems (ICDCS)*, Austin, May 1999.

[25] UCB/LBNL/VINT. Network simulator ns. `http://www.isi.edu/nsnam/ns/`, 2001.