CHAPTER 2

Related Work

A scientific theory should be as simple as possible, but no simpler.

— Albert Einstein (1879–1955)

The greatest challenge to any thinker is stating the problem in a way that will allow a solution.

— Bertrand Russell (1872–1970)

This chapter presents an overview of the research literature relevant for realistic traffic generation. We consider two types of works. First, we discuss the body of literature that developed the concepts and techniques currently in use for generating synthetic traffic in simulations and testbed experiments. Second, we examine the Internet measurement literature that informs the discussion of what is meant by "realistic" traffic generation. Intuitively, synthetic traffic resembling Internet traffic can only be realistic if derived from measurements conducted from real network links. We could argue that any Internet measurement paper helps to gain a better understanding of the nature of the Internet and its traffic, being therefore relevant for realistic traffic generation. However, the sheer size of the Internet measurement literature makes a complete overview impractical, so we will restrict ourselves to the main works that had a direct impact on Internet traffic generation. It is also interesting to note that the most recent trend in the field of traffic generation is precisely to combine traffic measurement and generation into a single, coherent approach [HCJS+01, LH02, SB04, HCSJ04].

Traffic generation for experimental networking research was identified as one of the key challenges in Internet modeling and simulation by Paxson and Floyd [PF95] in 1995. Interestingly, Floyd and Kohler [FK03] made a similar point in 2003, and argued that it was still difficult to conduct experiments with representative, validated synthetic traffic. While traffic measurement and Internet measurement in general have become increasingly popular in recent years, most studies are exploratory and provide little foundation to build traffic generators. This chapter provides an overview of the major works in the field

of Internet traffic generation, considering first packet-level traffic generation and then source-level traffic generation. Other aspects of traffic generation, such as load scaling, incorporating network-dependencies and implementation issues are discussed at the end of the chapter.

2.1 Packet-Level Traffic Generation

In this dissertation we restrict the question of generating realistic traffic to a single link. This is the most essential form of the traffic generation problem. It does not seem possible to tackle the problem of generating traffic for multiple links, say the backbone of an ISP, if single-link traffic generation is not fully understood.

The simplest way of generating realistic traffic on a single link is to inject packets into the network according to the characteristics of the packets observed traversing a real link. We will use the term *packet-level traffic generation* to refer to this approach. Packet-level traffic generation can mean either performing a *packet-level replay*, *i.e.*, reproducing the exact arrivals and sizes of every observed packet, or injecting packets in such a manner as to preserve some set of statistical properties considered fundamental, or relevant for a specific experiment. Packet-level replay, which has been implemented in tools like *tcpreplay* [tcpb], is a straightforward technique that is useful for certain types of experiments where configuration of the network is not expected to affect the generated traffic. In other words, whenever it is reasonable to generate traffic that is invariant of (*i.e.*, unresponsive to) the experimental conditions, then packet-level replay is an effective means for generating synthetic traffic. For example, packet-level replays of traces collected from the Internet have been used to evaluate cache replacement policies in routing tables [Jai90, Fel88, GcC02]. In this type of experiments, different cache replacement policies are compared by feeding the lookup cache of a routing engine with a packet trace and computing the achieved hit ratio. Also, studies that require malicious traffic generation can often make use of packet-level replay [SYB04, RDFS04]. Malicious traffic (*e.g.*, a SYN flood) is frequently not responsive to network conditions (and their degradation).

Before conducting an experiment in which traffic is generated using packet-level replay, researchers must obtain one or more traces of the arrivals of packets to a network link. These traces are collected using a packet "sniffer" to monitor the traffic traversing some given link. This packet capturing can be performed with and without hardware support. The most prominent example of software-only capture is the Berkeley Packet Filter (BPF) system [MJ93, tcpa]. BPF includes a packet capturing library,

libpcap, and a command-line interface and trace analysis tool, *tcpdump*. BPF relies on the promiscuous mode of network interfaces to observe packets traversing a network link and to create a trace of them in the "pcap" format. Due to privacy and size considerations, most traces only include the protocol headers (IP and TCP/UDP) of each packet and a timestamp of the packet's arrival. Monitoring high-speed links with a software-only system is problematic, given that traffic has to be forwarded from the network interface to the monitoring software using the system bus. The system bus may not be fast enough for this task depending on the load on the monitored link. High loads can result in "dropped" packets that are absent from the collected trace. Furthermore, the extra forwarding from the wire to the monitoring program, which usually involves buffering in the network interface and in operating system layers, makes timestamps rather inaccurate. In the case of BPF, timestamping inaccuracies of a few hundreds of microseconds are quite common. In order to overcome these difficulties, researchers often make use of specialized hardware that can extract headers and provide timestamps without the intervention of the operating system. This is of course far more expensive, but it dramatically improves timestamp accuracy and increases the volume of traffic that can be collected without drops. The DAG platform [Pro, GMP97, MDG01] is a good example of this approach, and it is widely used in network measurement projects. The timestamping accuracy of DAG traces is on the order of nanoseconds. Multiple DAG cards, possibly at different locations, can also be synchronized using an external clock signal, such as the one from the Global Positioning System (GPS). Besides collecting their own traces, researchers can also make use of public repositories of pcap and DAG traces, such as the Internet Traffic Archive [Int] and the PMA project at NLANR [nlab].

While packet-level replay is conceptually simple, it involves a number of engineering challenges. First, traffic generators usually rely on operating systems layers and abstractions, such as raw sockets, to perform the packet-level replay. Most operating systems provide no guarantee on the exact delay between the time of packet injection by the traffic generator and the time at which the packet leaves the network interface. Servicing interrupts, scheduling processes, *etc.*, can introduce arbitrary delays, which make the arrival process of the packet replay differ from the original and intended arrival process. This inaccuracy may or may not be significant for a given experiment. Another challenge is the replay of traces collected in high-speed links. The rate of packet arrivals in a trace can be far higher than the rate at which a single host can generate packets. For example, the speed at which a commodity PC can inject packets into the network is primarily limited by the speed of its bus and the bandwidth of its network interface. As a consequence, replying a high rate trace often requires an experimenter to partition the trace into subtraces that have to be replayed using a collection of hosts. In this case, it is

important to carefully synchronize the replay of these hosts. This is generally a difficult task, since the synchronization has to be done using the network itself, which introduces variable I/O delays. Clock drift is also a concern with common PC clocks.

Ye *et al.* [YVIB05] discussed packet-level replay of high rate traces, focusing on OC-48, and how to evaluate the accuracy of the replay. They proposed *flow-based splitting* to construct a partition of the original trace that can be accurately replayed by an ensemble of traffic generators. This addresses the challenge of replaying a trace using multiple traffic generators without reordering the packets within a flow. In contrast, round-robin assignment of packets to traffic generators, called *choice of N* in this work, results in packets belonging to the same flow generated by different traffic generators. As a consequence, the generated traffic exhibits substantial packet reordering. This reordering is due to the difficulty of maintaining the generators perfectly synchronized with commodity hardware, so one generator can easily get ahead of another and modify the order of packets within a flow. Ye *et al.* also discussed the difficulties created by buffering on the network cards, which modifies the properties of the packet arrival process at fine scales. An alternative to the approach in Ye *et al.* is to rely on specialized hardware. Most DAG cards support packet-level replay, bypassing the network stack. However, no information is available on how accurately the generated traffic preserves the properties of original packet arrival process.

Packet-level replay has two important shortcomings: it is inflexible and it is open-loop. Given that a packet-level replay is the exact reproduction of a collected trace, both in terms of packet arrival times and packet content, there is no way to introduce variability in the experiments other than acquiring a collection of traces and using a different trace in different runs of the experiments. This makes packet replay inflexible, since the researcher has to limit his experiments to the available traces and their characteristics. The "right" traces may not be available or may be difficult to collect. Even conducting experiments that study simple questions can be cumbersome. For example, a researcher that intends to test a cache replacement policy under heavy loads must find traces with high packet arrival rates, which may or may not be available. Similarly, evaluating a queuing mechanism under a range of (open-loop) loads requires one to find traces covering this range of loads, and may involve mixing traces from different locations, which could cast doubt on the realism of the resulting traffic and thus on the conclusions of the evaluation.

More flexible traffic generation can be achieved by generating packets according to a set of statistical properties derived from real measurements. The challenge then is to determine which properties of traffic are most important to reproduce so that the synthetic generated traffic makes the experiments

"realistic enough." For example, Internet traffic has been found to be very bursty, showing very frequent changes in throughput (both for packets and bytes per unit of time). Therefore, most experiments should make use of synthetic traffic that preserves this observed burstiness. Leland *et al.* [LTWW93] observed that this burstiness can be studied using the framework provided by *statistical self-similarity*. At a high-level, self-similarity means that traffic is equally bursty, *i.e.*, equal variance in arrival times, across a wide range of time scales. This is similar to the geometric self-similarity that fractals exhibit. Mathematically, statistical self-similarity manifests itself as *long-range dependence*, a sub-exponential decay of the autocorrelation of a time-series with scale. This is in sharp contrast to Poisson modeling and its short-range dependence, which implies an exponential decay of the autocorrelation with scale. Therefore, it is generally difficult to accept experimental results where synthetic traffic does not exhibit some degree of self-similarity. Accordingly, some experiments may simply rely on some method for generating a self-similar process [Pax97] and inject packets into the experiments according to this process. Studies on queuing dynamics, *e.g.*, [ENW96], made use of this traffic generation approach.

Other experiments with a more stringent need for realism may also attempt to reproduce other known properties of traffic. For example, a realistic distribution of IP addresses is essential for experiments in which route caching performance is evaluated. To accomplish this, packet-level traffic generation can be combined with a statistical model of packet arrival and a model of address structure. As one example, Aida and Abe [AA01] proposed a generative model based on the finding that the popularity of addresses follows a *powerlaw* (a heavy-tailed distribution with a hyperbolic shape). In contrast, Kohler *et al.* [KLPS02] focused on the hierarchical structure of addresses and prefixes, which is shown to be well-described by a multi-fractal model. Both studies could be used to enrich packet-level traffic generation.

2.2 Source-Level Traffic Generation

While packet-level traffic generation based on a set of statistical properties is convenient for the experimenter, and attractive from a mathematical point of view, it fails to preserve an essential property of Internet traffic. As Floyd and Paxson [PF95] point out, packet-level traffic generation is *open-loop*, in the sense that it does not preserve the feedback loop that exists between the sources of the traffic (the endpoints) and the network. This feedback loop comes from the fact that endpoints react to network conditions, and this reaction itself can change these conditions, and therefore trigger further changes in the behavior of the endpoints. For example, TCP traffic reacts to congestion by lowering its sending rate,

which in turn decreases congestion. A trace of packet arrivals collected at some given link is therefore specific to the characteristics of this link, the time of the tracing paths of the connections that traversed it, *etc.* Therefore, any changes that the experimenter makes to the experimental conditions make the packet-level traffic invalid since the traffic generation process is insensitive to these changes (unlike real Internet traffic). For example, packet-level replay of TCP traffic does not react to congestion in any manner.

The solution is to *model the sources* of traffic, *i.e.*, to model the network behavior of the applications running on the endpoints that communicate using network flows. Source-level models are then used to drive network stacks which do implement flow and congestion control mechanisms, and therefore react to changes in network conditions as real Internet endpoints do. As a result, the generated traffic is *closed-loop*, which is far more realistic for a wide range of experiments.

The simplest source-level model is the *infinite source model*. The starting point of the infinite source model is the availability of an infinite amount of data to be communicated from one endpoint to another. Generating traffic according to this model means that a traffic generator opens one or more transport connections, and constantly provides them with data to be transferred. This means that, for each connection, one of the endpoints is constantly writing (sending data packets) while the other endpoint is constantly reading (receiving data packets). The sources are never the bottleneck in this model. The only process that limits the rate at which the endpoints transmit data is the network, broadly defined to include any mechanism below the sources, such as TCP's maximum receiver window.

The infinite source model is very attractive for several reasons, which make it rather popular in both theoretical and experimental studies [FJ93b, KHR02, AKM04, SBDR05]. First, the infinite source model has no parameters and hence it is easy to understand and amenable to formal analysis. It was, for example, the foundation for the work on the mathematical analysis of steady-state TCP throughput [PFTK98, BHCKS04]. Second, its underlying assumption is that the largest flows on the network, which account for the majority of the packets and the bytes, "look like" infinite sources. For example, an infinite source provides a convenient approximation to a multi-gigabyte file download using FTP. Third, infinite sources are well-behaved, in the sense that, if driving TCP connections, they try to consume as much bandwidth as possible. They also result in the ideal case for bandwidth sharing. This makes them useful for experiments in the area of congestion control, since infinite sources can easily congest network links.

Despite their convenience, infinite sources are unrealistic and do not provide a solid foundation for

networking experiments, or even for understanding the behavior and performance of the Internet. The pioneering work by Cáceres *et al.* [CDJM91], published as early as 1991, provided a first insight into the substantial difference between infinite sources and real application traffic. These authors examined packet header traces from three sites (the University of California at Berkeley, the University of Southern California, and Bellcore in New Jersey) using the concept of application-level *conversations*. An application-level conversation was defined as the set of packets exchanged between two network endpoints. These conversations could include one or more "associations" (TCP connections and UDP streams). A general problem when studying traffic for extended periods is the need to separate traffic into independent units of activity, which in this case correspond to conversations. Endpoints may exchange traffic regularly, say every day, but that does not mean that they are engaged in the same conversation for days. Danzig *et al.* separated conversations between the same endpoints by identifying long periods without any traffic exchange, which are generally referred to as *idle times* or *quiet times* in the literature. In their study, they used a threshold of 20 minutes to differentiate between two conversations. The authors examined conversations from 13 different applications, characterizing them with the help of empirical cumulative distribution functions (empirical CDFs). The results include empirical CDFs for the number of bytes in each conversation, the directionality of the flow of data (*i.e.*, whether the two endpoints sent a similar amount of data), the distribution of packet sizes, the popularity of different networks, *etc.* Danzig and Jamin [DJ91] used these distributions in their traffic generation tool, *tcplib*. The results from this work are further discussed in Section 2.2.2.

Cáceres *et al.* pointed out a number of substantial differences between their results and the assumptions of earlier works. First, the majority of connections carried very small amounts of data, less than 10 KB in 75-90% of the cases. This is true for both interactive applications (*e.g.*, telnet and rlogin) and bulk transfer applications (*e.g.*, FTP, SMTP). This is in sharp contrast to the infinite availability of data to be transferred assumed in the infinite source model. The dynamics of such short data transfers are completely different from those of infinite sources, which for example have time to fully employ congestion control mechanisms. The second difference was that traffic from most applications was shown to be strongly bidirectional, and it included at least one request/response phase, *i.e.*, an alteration in the role of the endpoints as senders of data. The infinite source model is inherently unidirectional, with one of the endpoints always acting as the sender, and the other endpoint always acting as the receiver. Third, the authors observed a wide range of packet sizes, and a large fraction of the data packets were small, even for bulk applications. Data packets from an infinite source are necessarily full size, since there is by definition enough data to completely fill new packets.

These measurement results highlighted a substantial difference between infinite sources and real traffic, and later experimental studies demonstrated the perils of using traffic from infinite sources in the evaluating of network mechanisms. Joo *et al.* [JRF+99, JRF+01] demonstrated that infinite TCP sources tend to become synchronized, so they increase or decrease their transmission rate at the same time. This pattern is completely absent from more realistic experiments in which the majority of the sources have small and diverse amounts of data to send. As a result, loss patterns, queue lengths and other characteristics are strikingly different when more realistic synthetic traffic is used. Joo *et al.* also studied the difference between open-loop and closed-loop traffic generation.

The area of active queue management has provided several illustrations of the misleading results obtained with the unrealistic infinite sources. The first AQM scheme, RED, was presented by Floyd and Jacobson in [FJ93b], and evaluated using infinite sources. Their results showed that RED significantly outperformed FIFO, the usual router queuing mechanism. Later work by Christiansen *et al.* [CJOS00] demonstrated that RED offers very little benefit, if any, when exposed to more realistic traffic where sources are not infinite. In particular, they used a model of web-like traffic, which is discussed later in this chapter.

Paxson's analysis [Pax94] of packet header traces from seven different network links provided further support for the conclusions of Cáceres *et al.* In addition, Paxson considered the parsimonious modeling of traffic from different applications. He characterized four prominent applications, telnet, NNTP, SMTP and FTP, using analytic models to fit the empirical distributions. Analytic models are more commonly known as parametric models in the statistical literature, and correspond to classical distributions, such as the Pareto distribution, that can be fully characterized with a mathematical expression and only one or a few parameters. As Paxson pointed out, the use of analytic models results in a concise description of network applications that can be easily communicated and compared, and are often mathematically tractable. His methodology has had a lasting influence in application-level modeling. He clearly demonstrated that analytic fits (*i.e.*, parametric models) of the observed distributions can closely approximate the characteristics of real applications. However, it is important to remember that traffic is not necessarily more realistic when generated by analytic models as opposed to empirical models. Empirical CDFs, derived from network measurement of sufficient size, provide a perfectly valid foundation for traffic generators. Furthermore, finding analytic fits of complex random variables that do not match well-known statistical distributions is a daunting task.

2.2.1 Web Traffic Modeling

Modeling web traffic has received substantial attention since the sudden emergence of the World Wide Web in the mid-nineties. Arlitt and Williamson [AW95] proposed an early model for generating web traffic[1], based on packet header traces collected at the University of Saskatchewan. The model was centered around the concept of a conversation, as proposed by Cáceres *et al.* [CDJM91]. In this case, a conversation was the set of connections observed between a web browser and a web server. These authors were the first to consider questions such as the distribution of the number of bytes in requests and responses, the arrival rates of connections, *etc.* In general, the proposed model has parameters that are quite different from those of later works. For example, an Erlang model of response sizes was used, which is in sharp contrast to the heavy-tailness observed by other authors. While Arlitt and Williamson did not provide any details on the statistical methods they employed, it is likely that the small sample size (less than 10,000 TCP connections) made it difficult to develop a more statistically representative model.

One of the major efforts in the area of web traffic modeling oriented toward traffic generation took place at Boston University. Cunha *et al.* [CBC95] examined client traces collected by instrumenting browsers at the Department of Computer Science. Unlike the packet header traces used in Arlitt and Williamson, client traces include application information such as the exact URL of each web object requested and downloaded in each TCP connection. The authors made use of this information to study page and server popularity, which are relevant for web caching studies. In addition, the authors proposed the use of powerlaws for constructing a parametric model of web traffic. They relied on the Pareto distribution for modeling the sizes of downloaded objects, and the parameterless Zipf's law for modeling the popularity of specific pages. Crovella and Bestavros [CB96] used these findings to explain the long-range dependence observed in the packet arrivals of web traffic. Their explanation was derived from earlier work by Willinger *et al.* [WTSW97], which showed that the multiplexing of heavy-tailed ON/OFF sources results in long-range dependent traffic. Crovella and Bestavros demonstrated that the underlying distributions of web object sizes, the effects of caching and user preference in file transferring, the effect of user "think time", and the superimposition of many web transfers precisely creates the multiplexing process hypothesized by Willinger *et al.*

Crovella and Bestavros also showed that the explanation behind the suitability of powerlaws for describing the sizes of web objects is that the sizes of files are well described by powerlaws. This refined

[1] To be more specific, Arlitt and Williamson proposed a model of "Mosaic" traffic. Mosaic was the first web browser.

previous studies of file-system characteristics (*e.g.*, [BHK+91]), which observed long-tailed distributions of file sizes (but did not propose powerlaw models).

Powerlaw modeling has had a lasting impact on traffic modeling, which is natural given that the transfer of files is one of the most common uses of many application protocols. Countless studies have confirmed the usefulness of powerlaws for modeling application traffic. The eloquent term "mice and elephants" [GM01, MHCS02, EV03], often applied to Internet traffic, precisely refers to the basic characteristic of powerlaws: a majority of values are small (mice) but the uncommon large values (elephants) are so large that they account for a large fraction of the total value. For example, web traffic usually shows around 90% of web objects below 10 KB, but larger objects often account for 90% of the total bytes. Researchers have used this general finding of powerlaw sizes to develop a generic, and mostly ad hoc, source-level model. Traffic generated according to this model consists of a collection of TCP connections that transfer a single file, such that the distribution of file sizes follows a powerlaw. Researchers often refer to this kind of synthetic traffic as "mice-and-elephants-like" or "web-like" traffic [MGT00, KHR02]. This simple approach is rather convenient for traffic generation, but it ignores the more complex patterns of connection usage (*e.g.*, bidirectionality, quiet times, *etc.*), and the differences among applications present in real Internet traffic.

It is important to note that recent work on the characterization of web traffic has improved our understanding of powerlaw/heavy-tailed modeling. Downey revisited the modeling of file sizes in [Dow01b] and of flow sizes in [Dow01a], suggesting that lognormal distributions are more appropriate than powerlaws (or heavy-tailed distributions). The historical survey by Mitzenmacher [Mit04] uncovered similar controversies in other fields, such as economics and biology. Hernández-Campos *et al.* demonstrated that lognormal distributions and powerlaws offer similar results in the regions of the distribution for which enough samples are available, specifically in the body and in the "moderate" tail. Beyond these regions, in the "far" tail, the lack of samples makes it impossible to choose between different models. This is because, for a fixed set of parameters and a fixed sample size equal to the original number of observations, some samplings of the lognormal and the powerlaw models match the original distribution, while other samplings do not. Hernández-Campos *et al.* also proposed the use of a mixture model (*i.e.*, a combination of several classical models), the double Pareto lognormal, which enables far more accurate fits than those achieved with Pareto or lognormal models. The inherently more flexible double Pareto lognormal model can capture the systematic deviations from simpler models that are commonly observed in the tails of the distributions of web object sizes. Nuzman *et al.* [NSSW02] modeled HTTP connection arrivals using the biPareto distribution, which provides a simpler but powerful alternative to

mixture models. A Pareto distribution appears linear in a log-log scale, while the biPareto distribution shows two linear regions and a smooth transition between them. The biPareto distribution is therefore a generalization of the Pareto distribution.

The modeling efforts at Boston University culminated with the development of the SURGE model of web traffic [BC98]. The SURGE model described the behavior of each user as a sequence of web page downloads and think times between them. Each web page download consisted of one or more web objects downloaded from the same server. Barford and Crovella provided parametric fits for each of the components of the SURGE model, heavily relying on powerlaws and other long-tailed distributions. They also studied how SURGE traffic stressed web servers, and found SURGE's high burstiness far more demanding in terms of server CPU performance than that of less elaborate web traffic generators, such as the commercial WebStone.

A model of web traffic contemporary to SURGE was also presented by Mah [Mah97]. It described web traffic using empirical CDFs, which were derived from the analysis of packet header traces. As in the case of the SURGE model, the data came from the population of users in a computer science department. The two models were compared by Hernández-Campos *et al.* [HCJS03], showing substantial consistency.

The introduction of persistent connections in HTTP motivated further work on web traffic modeling. Barford *et al.* studied the performance implications of persistent connections [BC99], and modified the SURGE model to incorporate persistency [BBBC99]. The analysis of persistent connections was also a major topic in Smith *et al.* [SHCJO01] and Hernández-Campos *et al.* [HCJS03]. These studies were far larger in scope, focusing on the web traffic of an entire university rather than of a single department. These latter two works provided the starting point for the analysis method presented in this dissertation.

Many experimental studies made use of synthetic traffic generated according to one of the aforementioned web traffic models. For example, Christiansen *et al.* [CJOS00] made use of the Mah model, while Le *et al.* [LAJS03] used the Smith *et al.* model. The popular NS-2 [BEF+00] network simulator also supports web traffic generation using models that are structurally similar to the SURGE model. This feature of NS was used in Joo *et al.* [JRF+99, JRF+01] to compare web traffic and infinite sources, and by Feldmann *et al.* [FGHW99] to study the impact of different parameters of the web traffic model on the burstiness of the generated traffic. Another web traffic generator available in NS-2 was developed by Cao *et al.* [CCG+04]. Unlike other web traffic models, it was connection-oriented rather than user-oriented, and included non-source-level characteristics, such as packet sizes.

An important effort in web traffic analysis and generation was "Monkey See, Monkey Do" method,

developed by Cheng *et al.* [CHC+04a]. The method involved recording source-level and network-level characteristics for each observed connection, and reproducing these characteristics using a synthetic workload generator. This idea is similar to the one developed in this dissertation, although we tackle the modeling and generation of entire traffic mixes and not just web traffic. In addition, their measurement methods were optimized for monitoring traffic near Google's web servers. The authors assumed independent short flows, data acquisition close to well-provisioned web servers, and no congestion in the client-to-server direction (which was plausible in the context of requests that were far smaller than responses).

2.2.2 Non-Web Traffic Source-level Modeling

Two prominent source-level modeling efforts took place before the invention of the World Wide Web. Danzig and Jamin [DJ91] developed tcplib, a collection of source-level descriptions of traffic. It included descriptions of the following applications:

- Telnet was described using three random variables: connection duration, packet inter-arrival time, and packet size. The initiator of the Telnet connection always sent one-byte packets, while the acceptor responded with packets matching the packet size distribution. The authors claimed that rlogin connections were also well-described by this model.
- File Transfer Protocol (FTP) was described using three random variables: number of items transferred, item size (*i.e.*, file size), and packet size. The model only described FTP-DATA transactions used to transfer a single file or a directory listing. It did not describe the FTP-Control connection that each client/server pair must use to manage each FTP-DATA transaction.
- Simple Mail Transfer Protocol (SMTP) was described using only one random variable: item size, which included size of mail message and address verification (*i.e.*, control) messages. Responses from the acceptor were considered negligible, and not modeled.
- Network News Transfer Protocol (NNTP) was described using two random variables: number of items transferred, and size of items (*i.e.*, NNTP articles). The bidirectional nature of the protocol and the use of control messages was not part of the model.

Tcplib also included a model of phone conversations with two random variables, talk spurt duration and quiet time (*i.e.*, pause) duration, borrowed from [Bra65]. Each random variable was specified using an

empirical CDF. Traffic generation involved using the inverse transformation method [Jai91] to sample each empirical CDF independently.

In general, the application models in tcplib were rather simplistic, but they represented a giant step forward from the non-measurement-derived models of the early 90s. However, the use and capabilities of the modeled applications has dramatically changed since the development of tcplib. For example, the size of attachments in SMTP connections has dramatically increased due to the widespread implementation of Multipurpose Internet Mail Extensions (MIME). In addition, newer applications have become prominent or replaced the ones in tcplib. For example, the Telnet protocol has been mostly replaced by the Secure Shell (SSH) protocol. SSH is an encrypted protocol, so it requires more bytes per message. It also supports port forwarding, wherein other applications can communicate through SSH connections.

Paxson [Pax94] studied the same four applications as in tcplib, developing parametric models for each of them. Paxson also discussed how application characteristics change over time and across sites. This inherent variability motivated the use of parametric models, which are necessarily approximations of the empirical data. This approximation is not worse than the variability observed over time and across sites, so the author argued that parametric models were as accurate as empirical ones, but with the added benefits of being mathematically tractable and parsimonious. His analysis showed that bulk-transfer sizes were generally well-modeled by the log-normal distribution. Another of his findings was that connection inter-arrivals (except those of NNTP connections) were consistent with non-homogeneous Poisson arrivals, with fixed hourly rates.

The methodological contribution in Paxson's work is substantial. He demonstrated the difficulty of providing statistically valid parametric models of the distributions associated with Internet traffic. He consistently observed parametric fits that were clearly adequate when examined graphically, but that failed traditional goodness-of-fit tests. This was caused by the massive sample sizes, an endemic characteristics of traffic measurement datasets. As an alternative to the statistical tests, Paxson proposed the use of a goodness-of-fit metric, which provides a quantitative assessment of the distance between the empirical data and the parametric model. His proposed metric is however insensitive to deviations in the tails, casting doubt on the approach due to the ubiquitous finding of heavy-tailed phenomena in network traffic.

Web traffic quickly dominated most traffic mixes after its emergence in 1995, and remained the most prominent traffic type until file-sharing applications surpassed it in recent years. This motivated a large body of work on web traffic characterization, and little attention was paid to other traffic. The models

developed by Danzig, Jamin and Paxson, were not improved or updated by other researchers.

File-sharing applications currently rival or frequently surpass web traffic in terms of traffic volume. They also represent a harder modeling problem than web traffic. The number of file-sharing applications is large and they use widely different communication strategies. Furthermore, the set of popular file-sharing applications is constantly changing. There is a growing body of traffic modeling literature focusing on file-sharing applications, but no traffic generator is yet available. Two prominent modeling studies were conducted at the University of Washington. Sariou *et al.* [SGG02] studied Napster and Gnutella traffic, and Gummadi *et al.* [GDS+03] studied Kazaa traffic. Karagiannis *et al.* [KBBkc03] examined a larger set of file-sharing applications in backbone links.

Modeling of multimedia traffic has also received some attention. Variable bit-rate video was studied in Garret *et al.* and Knightly *et al.* [GW94, KZ97]. Real Audio traffic was studied by Mena and Heidemann [MH00], providing a first source-level view of streaming-media, mostly on UDP flows.

There are commercial synthetic traffic generation products such as Chariot [Inc] and IXIA but these generators are typically based on a limited number of application source types. Moreover, it is not clear that any are based on empirical measurements of Internet traffic.

2.2.3 Beyond Single Application Modeling

The need for more representative traffic generation has motivated research on methods that can tackle the modeling of the entire suite of applications using an Internet link. The work in this dissertation lies in this area. Our preliminary steps were an extension of the methods used to model web traffic in Smith *et al.* [SHCJO01] to model other applications, as described in Hernández-Campos *et al.* [HCJS+01]. The same kind of analysis of TCP header sequence numbers, acknowledgment numbers and connection quiet times applied to web traffic was used to populate models of SMTP and NNTP traffic. These models were derived from packet header traces collected at the University of North Carolina at Chapel Hill, and consisted of empirical distributions capturing different source-level characteristics of these protocols, such as object sizes. Lan and Heidemann [LH02] conducted a related effort, reusing the same techniques and software tools for data acquisition. Their RAMP tool populated models of web and FTP traffic directly from packet header traces, and generate traffic accordingly.

Harpoon [SB04] also tackled the same problem that is the focus of this dissertation. They considered the problem of analyzing entire traffic mixes and generating traffic accordingly. Their measurement

methods were far less elaborate. Rather than the detailed models of the ADU exchange in TCP connections used in our work, Harpoon focused on modeling flows. Flows are defined as sets of packets with the same source and the same destination. As a consequence, Harpoon modeled each TCP connection as two unidirectional flows. Another difference with our approach is that Harpoon did not incorporate the notion of bidirectional data exchange, neither sequential nor concurrent, essentially treating multiple ADUs (as defined in the a-b-t model) as a single ADU. Idle times within connections were not part of the Harpoon traffic model either. In addition, any measured flow (*i.e.*, one side of a connection) with only a small amount of data or with only acknowledgment packets was not used for traffic generation. This substantially simplified the modeling, but it eliminated the rich packet-level dynamics observed in TCP connections, and demonstrated in later chapters of this dissertation. In addition to this, network-level parameters were not part of the data acquisition, so round-trip times and maximum receiver window sizes were arbitrarily chosen. Harpoon could also generate UDP traffic. The underlying model was to send packets at a constant bit rate, with either fixed or exponentially distributed interval arrivals. These models were not populated from measurement. Another novel feature of Harpoon was the ability to generate traffic that reproduced IP address structure according to a measured distribution of address frequency. Their study included a comparison between Harpoon's closed-loop traffic and traffic from a commercial (open-loop) packet-level traffic generator, demonstrating substantial differences. For example, closed-loop sources were shown to back off as congestion increases, while open-loop source did not. Like the work in this dissertation and Lan and Heidemann, Harpoon provided an automated method to acquire data and use it to generate traffic, which Sommers and Barford eloquently called "self-tuning" traffic generation. We could say that there is a growing consensus in the field of traffic generation regarding the need to develop tools that combine measurement and generation to tackle the wide variability over time and across links found in real Internet traffic.

2.3 Scaling Offered Load

One of the key requirements of traffic generation is the ability to *scale the offered load*, *i.e.*, to generate a wide range of link loads with the same model of application behavior. This makes it possible to evaluate the performance of a network mechanism under various loads, which translates into different degrees of congestion, while preserving the same application mix. For example, the evaluation of AQM mechanism in [CJOS00, LAJS03] compared the performance of FIFO to RED and other AQM mechanisms for loads between 50% to 110% of a link's capacity where the queuing mechanism was used. In these studies,

the authors preceded their study by a set of *calibration* experiments. These experiments were used to derive an expression for the linear dependency between the number of (web) user equivalents and the average offered load, which enabled the researchers to systematically scale offered loads in their evaluation experiments. Calibration is generally applicable to any application-level model. When calibrating, the researchers try to relate one or more parameters of the model and the average offered load to obtain a *calibration function*. Deriving a calibration function is a time-consuming process, since an entire collection of experiments must be run to correlate offered load and model parameters with confidence.

Kamath *et al.* [KcLH+02] studied load scaling methods, but they concentrated only on scaling up the offered load. Their intention was to conduct experiments with much higher offered loads than those observed during measurement. In particular, they considered the problem of generating traffic for loading a 1 Gbps link using only measurements from a 10 Mbps link, an 11-hour packet header trace. The authors considered three different techniques. The first two techniques involved a transformation of the original trace into a scaled-up version, and then a packet-level replay. The first transformation technique was *packet arrival scaling*, which scales up the load by multiplying the arrival time of each packet in the original trace by a constant factor between 0 and 1 (*i.e.*,,shrinking packet inter-arrivals). In their study, they used a scaling factor of 0.001. The second transformation technique is *trace merging*, which scales up load by merging, *i.e.*, superimposing, the packet arrivals from more than one trace. They divided the 11-hour trace into 100 subtraces and then combined them to form a shorter, higher-throughput trace. The third technique is *structural modeling* which meant to develop a web traffic model from the original trace using the methods in Smith *et al.* [SHCJO01]. The authors did not discuss how the load created by this structural model was increased. Their analysis compared a number of distributions from the generated traces to those from the original trace. Packet arrival scaling was shown to completely distort flow durations and destination address diversity. Trace merging reproduced flow and packet arrival properties accurately, but it distorted destination address characteristics (studied using the number of unique addresses observed per unit of time). Web traffic generation was accurate, but it showed far less complex distributions of connection bytes, packet sizes, and connection durations. This is because a structural model based only on web traffic lacks the diversity of application behavior, and therefore communication patterns, in the original trace, which included traffic from many different applications and not just web traffic.

2.4 Implementing Traffic Generation

Source-level traffic generators for network testbeds (rather than for software simulators) are usually implemented using user-level programs that make use of the socket interface to generate traffic. This is the case for *tcplib* [DJ91], *httperf* [MJ98], SURGE [BC98], and other web traffic generators [BD99, CJOS00]. In order to introduce network-level parameters in test-bed experiments, such as a realistic distribution of round-trip times, it is necessary to rely on a layer of simulation either in the end hosts or somewhere in the path of the traffic. For example, Rizzo's *dummynet* [Riz97] makes it possible to apply arbitrary delays, loss rates and bandwidth constraints on the end systems to specific network flows or collections of network flows (that share a network prefix). The implementation combines event-driven simulation and packet queuing, and sits between the IP and link layers. Dummynet is part of the standard distribution of the FreeBSD operating system. The experiments in this dissertation were performed using an extended version of dummynet that can be controlled from the application layer[2].

Kamath *et al.* [KcLH+02] argue that source-level traffic generation is much more demanding in terms of CPU and memory processing than packet-level replay. While it is indeed true that far more CPU time is needed to simulate endpoint behavior and use network stacks, memory requirements are actually far more stringent for packet-level replay. This is because packet header traces are much longer than their source-level representations. For example, the approach in this dissertation considers the replay of source-level traces that are roughly 100 times smaller than the packet header traces from which they were derived.

2.5 Summary

Our review of related work has focused on the existing literature in network traffic generation, including works relevant for data acquisition and traffic modeling. Characterizing network traffic at the packet level provides important insights, such as the finding of pervasive self-similarity by Willinger *et al.* [WTSW97]. However, this approach does not provide the proper foundation for generating traffic for most experimental studies. As argued by Floyd and Paxson [PF95], packet-level traffic generation breaks the end-to-end feedback loop in adaptive network protocols, such as TCP, resulting in traffic that does not react to the experimental conditions realistically. On the contrary, source-level models enable

[2]This is also possible in the original implementation, using one firewall rule for each flow, but it does not scale to the hundreds of simultaneous flows in our experiments.

closed-loop traffic generation, so they are applicable to a wider range of situations.

In the past, source-level traffic generation has been associated with models of application behavior. Our overview of the state-of-the-art discussed several highly influential works devoted to application-level modeling. Cáceres *et al.* [CDJM91] introduced empirical application models to networking research. Paxson [Pax94] proposed the use of more statistically rigorous methods for developing parametric source-level models. Crovella *et al.* [CB96] developed a rich model of web traffic, and explained self-similarity in terms of source-level characteristics.

Application-level modeling has some important shortcomings that provide the motivation for this dissertation. Internet traffic mixes are created by a large number of distinct applications, so single application models are not representative of real traffic. Furthermore, the composition of traffic mixes is constantly changing, and even individual applications often evolve, modifying the way in which they interact with the network. As a consequence, the number of high-quality application-level models is small (and insufficient), and these models are hardly ever updated. In this dissertation, we propose a more scalable approach to source-level modeling, where application behavior is described in a generic, but still detailed, manner. Furthermore, our data acquisition methods are efficient and mostly automated, dramatically reducing the time to go from measurement to traffic generation.

Our combination of data acquisition and traffic generation is most closely related to two contemporary works. Sommers and Barford [SB04] developed the Harpoon approach for generating traffic mixes whose characteristics are derived from measurements in an algorithmic manner. Their approach did not include any detailed source-level modeling of TCP connections. They described a connection simply as a unidirectional file transfer whose size is equal to the total amount of payload in its packets. In contrast, our primary emphasis is on detailed source-level modeling, where we introduce the a-b-t model and uncover the dichotomy between sequential and concurrent data exchange. Harpoon made use of simplified network-level parameters, which are set to arbitrary constants. In our approach, network-level parameters are carefully measured and incorporated into the traffic generation. The work by Sommers and Barford considered two issues that are not addressed in our own work. First, they proposed a method for generating UDP traffic. The underlying source-level model is however not derived from measurement. Second, they reproduced the IP address distribution in the replayed trace. This cannot be performed with publicly available traces, like ours, since they are anonymized.

Another work similar to ours is Cheng *et al.* [CHC+04a]. The authors presented a method for characterizing packet header traces of web traffic and accurately replaying them. Generated traffic was

evaluated by comparing the original trace with its synthetic version generated in a testbed. We tackle the same source-level trace replay problem but applied to every application rather than only to web traffic. Our approach is more ambitious and necessarily more abstract.

Our work also considers the common problems of resampling and scaling traffic load in networking experiments. In general, scaling offered load has been performed by conducting a preliminary experimental study to relate the parameters of the source-level model and the offered load. For example, Christiansen *et al.* [CJOS00] computed a calibration function that described offered load as a function of the number of user equivalents employed in web traffic generation. We propose an alternative approach that eliminates the need for preliminary calibration studies.