CHAPTER 7

Trace Resampling and Load Scaling

That which is static and repetitive is boring. That which is dynamic and random is confusing. In between lies art.

— John A. Locke (1632–1704)

Everything that can be counted does not necessarily count; everything that counts cannot necessarily be counted.

— Albert Einstein (1879–1955)

The previous chapters presented a complete methodology for reproducing the traffic observed on a network link in a closed-loop manner, and proposed a number of metrics for studying the realism of the generated traffic. In this chapter, we study ways to introduce statistical variability in synthetic traffic in a meaningful and controlled manner. In addition, we address the need for changing offered load in network experiments. The methods that we introduce in this chapter add significant flexibility to our traffic generation approach, enabling researchers to perform a wider range of experiments.

In the approach presented so far, traffic is generated according to a trace $\mathcal{T}_c = \{(T_i, C_i)\}$. Each augmented connection vector C_i is replayed starting at time T_i. This implies that two different replays of $\mathcal{T}_c$ using the same hardware and the same physical network result in very similar synthetic traffic. In both cases, the synthetic traffic has the same number of TCP connections, replaying the same source-level behaviors under the same network-level parameters, and starting exactly at the same times. Only tiny variations would be introduced on the end-systems by changes in clock synchronization, operating system scheduling and interrupt handling, and at switches and routers by the stochastic nature of packet multiplexing. This reproducibility was exactly what was needed to evaluate how well synthetic traffic approximated the real traffic from which it derived.

However, in the practice of experimental networking, experimenters often want to introduce more variability in their experiments. One way of accomplishing this is to use more than one trace in a

replay, exposing the studied network protocol or mechanism to different types of workloads. This is highly desirable, but it has its drawbacks. First, the experimenter may want to perform a number of experiments that is larger than the number of available traces. Second, traces from different sites, and even traces from the same site but collected at different times of the day, may be so different that it becomes difficult to extrapolate from the results of the experiments.

A different, and complementary, approach is to conduct several experiments using traffic that "looks like" some specific trace $\mathcal{T}_c$, without exactly replaying $\mathcal{T}_c$ over and over. The first challenge in devising a method for accomplishing this task is to define what "looks like" mean. This involves creating a model (either formal or informal) of the traffic which is general enough to contain $\mathcal{T}_c$ but specific enough to always *resemble* the original trace. Running different experiments then requires to instantiate this model several times to create new derived traces $\mathcal{T}_c'$, $\mathcal{T}_c''$, ... and to generate traffic with these new traces using their source-level trace replay. In this chapter, this instantiation consists of resampling the set of connection vectors in $\mathcal{T}_c$ and assigning them new start times. Statistical variability in the derived traces comes from the resampling of the original connection vectors, and from the process of connection start times. We preserve the statistical properties of the original set of connection vectors by resampling *entire* connection vectors, *i.e.*, we do not manipulate the sizes and order of the ADUs and inter-ADU quiet times inside connection vectors. Our belief is that a trace created by modifying the source-level behavior of the connection vectors or their network-level parameters "does not look like" the original trace. For example, doubling the size of the ADUs in $\mathcal{T}_c$ is an easy way of creating a new trace and increasing the offered load. However, the resulting connection vectors have little to do with the connections observed in the link from which $\mathcal{T}_c$ was collected. Our choice to maintain connection vectors intact is reasonable, and consistent with the spirit of our overall methodology, which goes to great lengths to accurately characterize the source-level characteristics of each connection. Other researchers may have a different mental model of the legitimate level of statistical variability which could be introduced in $\mathcal{T}_c', \mathcal{T}_c'', \ldots$ We propose a specific solution and demonstrate that it is reasonable using quantitative data. A discussion of the different philosophies is outside the scope of this work.

The two sections in this chapter describe two techniques for introducing variability in the source-level replay of a trace. Section 7.1 describes *Poisson Resampling*. This technique assumes that connections are independent of each other, which is a reasonable choice for highly aggregated traffic. Poisson Resampling involves randomly resampling the connection vectors in $\mathcal{T}_c$ in an independent manner to create a new $\mathcal{T}_c'$. New start times are given to each resampled connection vector in a such a way that connection inter-arrivals are exponentially distributed. As we will show, empirical data support the choice of exponential

inter-arrivals.

Section 7.2 describes *Block Resampling*. This technique involves resampling blocks of connection vectors, preserving arrival dependencies among the connections inside the same block. Each block is the set of connections observed in an interval of fixed duration (*e.g.*, 1 minute) in the original trace. We will demonstrate that this technique, unlike Poisson Resampling, preserves the long-range dependence in the connection arrival process found in real traces. This cannot be achieved by sampling independently from an exponential (or any other) distribution. Note that we will reserve the term *resampling* for randomly selecting connection vectors, and the term *sampling* for randomly drawing values from a parametric distribution, such as the exponential distribution.

The second topic of this chapter is how to manipulate a trace $\mathcal{T}_c$ to modify the traffic load (*i.e.*, average byte throughput) that this trace offers during its source-level replay. This is a common need in experimental networking research, where the performance of a network mechanism or protocol is often affected by the amount of traffic to which it is exposed. For example, active queue management mechanisms have very different performance depending on the level of utilization of the input link, so researchers generally perform experiments with offered loads that range from 50% to 120% of the link bandwidth. Rather than trying to find or collect traces with the exact range of loads needed (which is generally difficult), we propose to produce a collection of resampled traces with the intended range of offered loads.

Average load l is defined as the total number of bytes injected in the network s divided by the total duration of the experiment d. Changing the average load in an experiment of constant duration therefore implies creating a *scaled trace* $\mathcal{T}_c'$ with a higher or a lower total number of bytes. Once again, the assumption is that it is possible to create a scaled trace $\mathcal{T}_c'$ which "looks like" the original $\mathcal{T}_c$ but with a larger or smaller number of bytes. This requires a model of traffic that is general enough to encompass $\mathcal{T}_c$ and traces derived from $\mathcal{T}_c$ with different offered loads. As should be obvious, the problems of introducing statistical variability and changing average load are related, and can naturally be treated together, as we will do in this chapter. The two techniques mentioned above, Poisson Resampling and Block Resampling, provide the foundation for deriving scaled traces. In both cases, the resampling of $\mathcal{T}_c$ to create a scaled $\mathcal{T}_c'$ can be modified to achieve a target average load. This means that our scaling method is *predictable*, which is an advance over earlier traffic generation methods, *e.g.*, [CJOS00, LAJS03, SB04]. These earlier methods required a separate experimental study, a *calibration*, to construct a function coupling the parameters of the traffic generator and the achieved load. For example, web traffic generators usually

require a calibration to discover the relationship between average load and the number of user equivalents. The scaling methods presented in this chapter eliminate the need for this extra study. Their starting point is the observation that the average load offered by the source-level replay of $\mathcal{T}_c$ is a deterministic function of the total number of bytes in the ADUs of $\mathcal{T}_c$. We will show that these observation holds true using numerical simulations and testbed experiments. In contrast, the same analysis will demonstrate that the average load offered by the replay of $\mathcal{T}_c$ is not strongly correlated with its number of connections. In the case of Poisson Resampling, our method to construct a new trace $\mathcal{T}_c'$ with a specific target offered load involves resampling $\mathcal{T}_c$ until the desired total number of bytes (coming from ADUs) is reached. In the case of Block Resampling, constructing a new trace $\mathcal{T}_c'$ with a specific target offered load involves subsampling blocks ("thinning") to decrease load, or combining two or more blocks ("thickening") to increase load.

7.1 Poisson Resampling

7.1.1 Basic Poisson Resampling

The first technique we consider for introducing variability in the traffic generation process is *Poisson Resampling*. The starting point of every method presented in this chapter is a connection vector trace $\mathcal{T}_c = \{(T_i, C_i) \,|\, i = 1, 2, \ldots, n\}$ where C_i is an augmented connection vector (an a-b-t connection vector plus some network-level parameters), and T_i is its relative start time. The basic version of our Poisson Resampling technique consists of deriving a new trace $\mathcal{T}_c' = \{(T_j', C_j') \,|\, i = 1, 2, \ldots, n'\}$ by randomly choosing connection vectors from $\mathcal{T}_c$ without replacement, and assigning them start times according to an exponential distribution. We define the duration d of $\mathcal{T}_c$ as $T_n - T_1$, the length of the interval in which connections are started[1]. Given a target duration d' for $\mathcal{T}_c'$, the Poisson Resampling algorithm iteratively adds a new (T_j', C_j') to $\mathcal{T}_c'$ until $T_j' > d'$. Each C_j' is equal to some randomly selected C_i, and

$$T_j' = T_{j-1}' + \delta_j,$$

where δ_j is sampled independently from an exponential distribution. The mean μ' of this exponential distribution provides a way to control the *density* of connections in the derived trace. For example, if we intend to have the same density of connections in $\mathcal{T}_c'$ as in $\mathcal{T}_c$, we can compute the mean inter-arrival

[1]This duration is always slightly below the true duration of the original packet header trace, since at least the packets of the last connection started are observed after its start time.

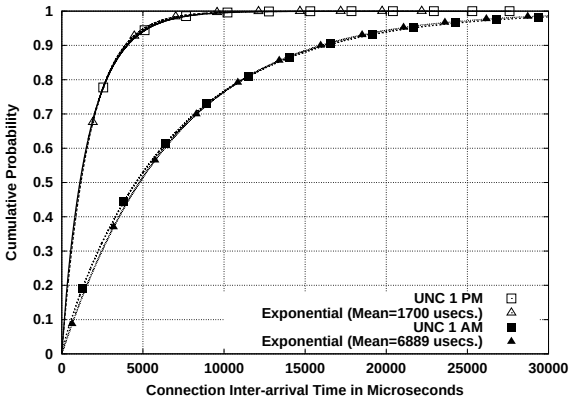

Figure 7.1: Bodies of the distributions of connection inter-arrivals for UNC 1 PM and 1 AM, and their exponential fits.

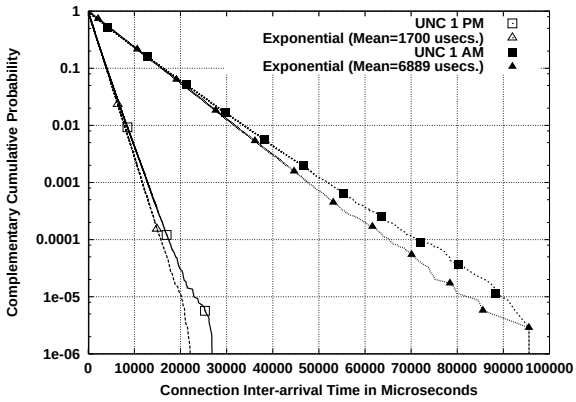

Figure 7.2: Tails of the distributions of connection inter-arrivals for UNC 1 PM and 1 AM, and their exponential fits.

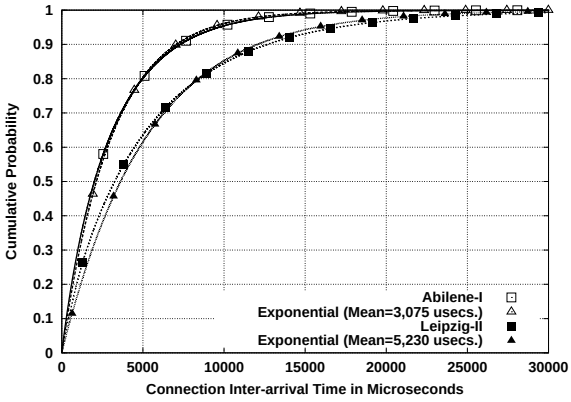

Figure 7.3: Bodies of the distributions of connection inter-arrivals for Abilene-I and Leipzig-II, and their exponential fits.

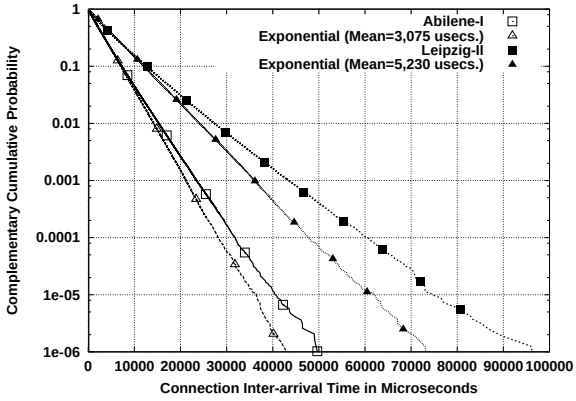

Figure 7.4: Tails of the distributions of connection inter-arrivals for Abilene-I and Leipzig-II, and their exponential fits.

time $\mu = d/n$ of the connection vectors in $\mathcal{T}_c$, and use it as the mean μ' of the experimental distribution used to construct $\mathcal{T}_c'$. Given the light tail of the exponential distribution, the resulting number n' of connection vectors in $\mathcal{T}_c'$ is always very close to d'/μ'. If $d = d'$, the number of connection vectors in $\mathcal{T}_c'$ is also very close to n.

The resampling technique described above has the advantage of its simplicity. Furthermore, it is statistically appealing, since the exponential distribution naturally arises from the combination of independent events. The use of connection inter-arrivals sampled independently from an exponential distribution is intuitively consistent with the view of traffic as a superposition of a large number of independent connections that transmit data through the same network link. Our empirical data, presented in Figures 7.1 to 7.4, confirm the applicability of this inter-arrival model. Figure 7.1 shows two pairs

of CDFs comparing real connection arrivals and their exponential fits. The first pair (white symbols) corresponds to the distribution of connection inter-arrivals for UNC 1 PM (squares), and an exponential distribution[2] with the same mean (triangles). The second pair (black symbols) shows the distribution of connection inter-arrivals for UNC 1 AM and an exponential distribution with the same mean. Both fits are excellent, so exponentially distributed connection inter-arrivals are clearly a good starting point for a trace resampling technique. The tails of the empirical distributions, shown in Figure 7.2, are also consistent with the fitted exponentials. Their slope is slightly lower, which could motivate a fit with a more general distribution like Weibull. However, a small improvement in the fit would require an increase in the complexity of the model, since the one-parameter exponential model would have to be replaced by the two-parameter Weibull model. This additional effort would produce only a limited gain given that the exponential fit is excellent for 99.9% of the distribution.

Figures 7.3 and 7.4 consider another two traces, Abilene-I and Leipzig-II. The bodies are again very closely approximated, but the tails are heavier for the original data. Note that this effect is more pronounced as the traces get longer. The duration of the UNC traces is one hour, the duration of Abilene-I is 2 hours, and the duration of Leipzig-II is 2 hours and 45 minutes. This could suggest that the worse fit is due to non-stationarity in the connection arrival process, which becomes more likely for longer traces. Further analysis is needed to confirm this hypothesis or find an alternative explanation. We must note that these results are in sharp contrast with those in Feldmann [Fel00], where the empirical inter-arrival distributions were significantly different from the bodies[3] of fitted exponential distributions. The reason for this difference is unclear at this point[4].

The main problem with the basic Poisson Resampling technique is the lack of control over the load offered by the replay of $\mathcal{T}_c'$. As we will demonstrate, the number of connections in $\mathcal{T}_c'$ is only loosely correlated to the offered load. As a consequence, it becomes difficult to predict the load that a Poisson resampled trace generates, even for resamplings of the same duration and mean inter-arrival rate. This would force researchers to create many resampled traces until they hit upon a resampling with the intended offered load. We studied the wide range of offered loads that result from basic Poisson Resampling by performing a large of number of resamplings of the same connection vector trace $\mathcal{T}_c$.

As discussed in the introduction of this chapter, average load l created by $\mathcal{T}_h$ is equal to the total number of bytes s in $\mathcal{T}_h$ divided by its duration d. Given that TCP headers and retransmitted segments

[2]The shown exponential distribution comes from randomly sampling the theoretical distribution $n-1$ times.

[3]The tails were not studied in that paper.

[4]Besides problems with the fitting or the data acquisition in the paper, we conjecture that this could be due to the slightly different type of data we considered in our study. Our connections were fully captured and included only those connections that actually carried data. Those in [Fel00] included degenerate cases in which no data was transferred.

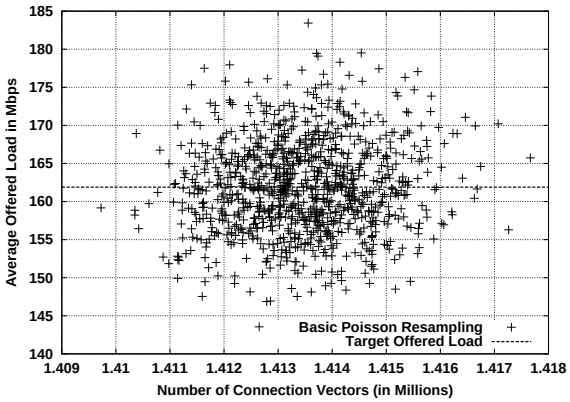

Figure 7.5: Average offered load *vs.* number of connections for 1,000 Poisson resamplings of UNC 1 PM.

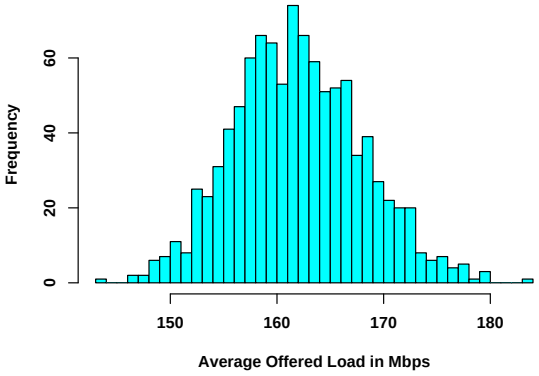

Figure 7.6: Histogram of the average offered loads in 1,000 Poisson resamplings of UNC 1 PM.

usually represent only a small fraction of s, the total size of the ADUs in $\mathcal{T}_c$ divided by d is also a close approximation of l. We use this approximation to examine the average loads offered by a large number of Poisson resamplings, considering the offered load of a resampling $\mathcal{T}_c'$ equal to the total size s' of its ADUs divided by its duration d'. It is also important to note that the traces we consider are bidirectional, and they do not necessarily create the same average load in both directions. The analysis in the rest of this section will focus only on one direction of the trace, the *target* direction, whose average load is given the total size of the ADUs flowing in that direction divided by the duration of the trace. More formally, the total size of the ADUs in the target direction is equal to

$$s' = \sum_{i \in C_{init}} \sum_{j=1}^{n_a^i} a_j^i + \sum_{i \in C_{acc}} \sum_{j=1}^{n_b^i} b_j^i, \tag{7.1}$$

where C_{init} is the set of connection vectors in $\mathcal{T}_c'$ initiated in the target direction, C_{acc} is the rest of the connection in $\mathcal{T}_c'$ (the connections accepted rather than initiated in the target direction), n_a^i and n_b^i are the numbers of a-type and b-type ADUs of i-th connection vector respectively, and a_j^i and b_j^i are the sizes of the j-th a-type and b-type ADU of the i-th connection vector respectively. Computing offered load as s'/d' is only a convenient (and reasonable) approximation of the load generated by replaying $\mathcal{T}_c'$. First, s' is an underestimation, since it does not take into account the total size of packet headers (only ADUs), and the retransmissions in the replay. Second, the duration of the replay of the connection vectors in $\mathcal{T}_c'$ will be somewhat above d'. d' only considers the period in which connections are started, but some of them will terminated after the last connection is started. An obvious example is the last connection. As we will demonstrate using experiments, the inaccuracy of s'/d' is very small, so it provides a good foundation for understanding load scaling. This calculation is obviously much more convenient than

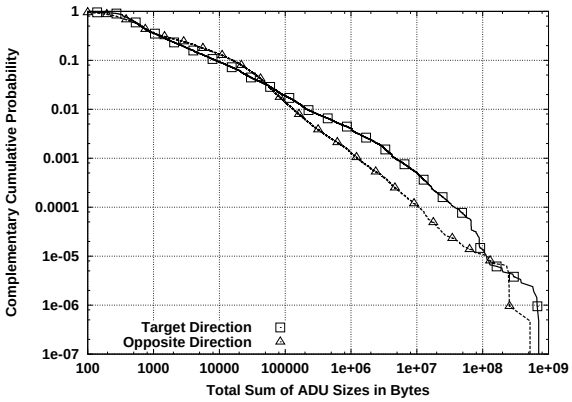

Figure 7.7: Tails of the distributions of connection sizes for UNC 1 PM.

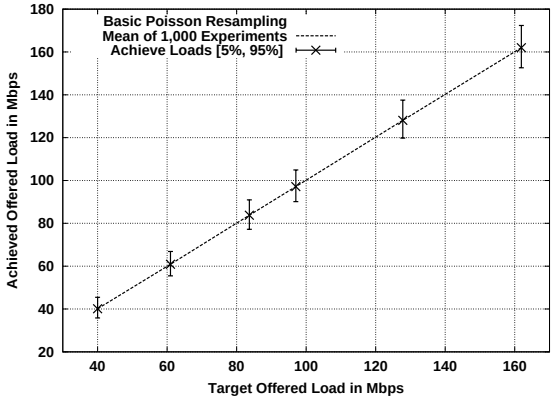

Figure 7.8: Analysis of the accuracy of connection-driven Poisson Resampling from 6,000 resamplings of UNC 1 PM (1,000 for each target offered load).

replaying thousands of resamplings in the testbed network.

Figure 7.5 shows a scatterplot of the results of 1,000 resamplings of UNC 1 PM. The duration of the resamplings and their mean rate of connection inter-arrival were equal to the ones in UNC 1 PM. For each resampling, the total number of connections is shown on the x-axis, while the resulting offered load s'/d' is shown on the y-axis. This plot demonstrates that basic Poisson Resampling results in traces with very small variability in the number of connection vectors, between 1,409,727 and 1,417,664 (the standard deviation σ was equal to 1,191.71). On the contrary, the range of offered loads is very wide, between 143.55 and 183.44 Mbps ($\sigma = 6.01$ Mbps), centered around the offered load of $\mathcal{T}_c$, 161.89 Mbps. The distribution of offered loads and its spread is further illustrated by the histogram in Figure 7.6.

The wide range of offered loads that can result from Poisson Resampling is due to the heavy-tailed nature of the distribution of the total number of bytes contributed by each connection vector. The tail of this distribution for the UNC 1 PM trace is shown in Figure 7.7. The values in the plot correspond to the target direction, *i.e.*, $\sum_{j=1}^{n_a^i} a_j$ for each connection in C_{init} and $\sum_{j=1}^{n_b^i} b_j$ for each connection in C_{acc}. The tails show non-negligible probabilities for very large sizes, and a linear decay of the probability over six orders of magnitude in the log-log CCDF. As a consequence, the contribution to the offered load made by each connection is highly variable and thus the number of connections in a trace is a poor predictor of its offered load. This makes basic Poisson Resampling inadequate for controlling load. Its only parameter is the mean inter-arrival rate of connections. This rate controls the same number of connections in the resampling, but not the total size of these connections, which varies greatly due to the heavy-tailed connection sizes. Figure 7.8 further illustrates this point using six sets of 1,000 trace

resamplings, each set with a different target offered load. The plot shows a cross marking the mean of the load achieved by each of the sets of 1,000 experiments. The variability in the offered loads is illustrated using error bars for each set of experiments. The lower and upper endpoints of the error bars correspond to the 5th and 95th percentiles of the loads offered by each set of trace resamplings. Each set of trace resamplings had a fixed mean inter-arrival rate μ'. Under the assumption that the mean offered load l is proportional to the number of connections n, we would expect the load to be inversely proportional to the mean arrival rate μ, since $\mu = d/n$. Therefore, if the resamplings have the same duration d, we would expect to achieve an offered load of

$$l' = \frac{l\mu'}{\mu} \tag{7.2}$$

in each resampling. This expected value is shown in the x-axis as "target offered load". The mean of the loads offered by each set of resamplings is indeed equal to the expected (target) value. However, the error bars show a wide range of offered loads for the same μ', which is undesirable for a load scaling technique. For example, the width of the range of loads offered by the resamplings with the highest target load was 20 Mbps. Differences of this magnitude between resamplings can have a dramatic effect on the experiments, completely obscuring differences among competing mechanisms. The difficulty of precisely controlling the load using only the number of connections as a parameter motivated our refinement of Poisson Resampling to achieve far more predictable offered loads.

7.1.2 Byte-Driven Poisson Resampling

As the previous section demonstrated, the number of connection vectors in a trace is a poor predictor of the mean offered load achieved during the replay of a resampled trace $\mathcal{T}_c'$. Therefore, controlling the number of connections in a resampling does not provide a good way of achieving a target offered load, and an alternative method is needed. In the idealized model of offered load in the previous section, the offered load l was said to be exactly equal to s/d. If so, we need to control the total number of bytes in the resampled trace $\mathcal{T}_c'$ to precisely match a target offered load. In *Byte-driven Poisson Resampling*, the mean inter-arrival rate of $\mathcal{T}_c'$ is not computed a priori using Equation 7.2. Instead, the target load l' is used to compute a target size $s' = l'd'$ for the payloads in the scaled direction.

Byte-driven Poisson Resampling has two steps:

1. We construct a set of connection vectors (without arrival times) by iteratively resampling the

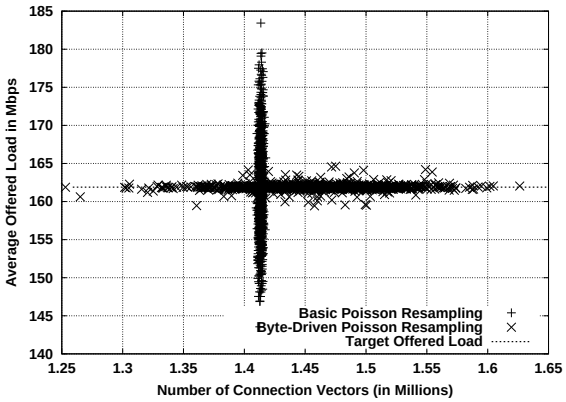

Figure 7.9: Comparison of average offered load *vs.* number of connections for 1,000 connection-driven Poisson resamplings and 1,000 byte-driven Poisson resamplings of UNC 1 PM.

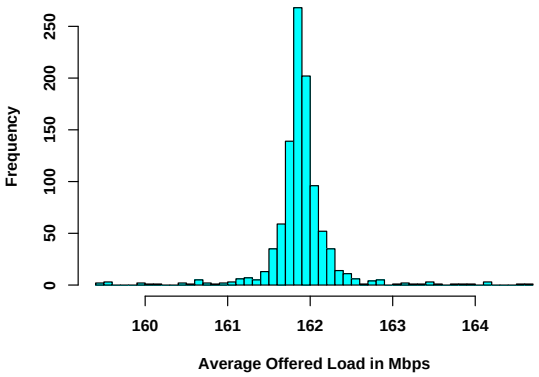

Figure 7.10: Histogram of the average offered loads in 1,000 byte-driven Poisson resamplings of UNC 1 PM.

connection vectors in $\mathcal{T}_c$ until the total payload size of the chosen connection vectors, computed using Equation 7.1, reaches s'.

2. We assign start times to the connection vectors in the resampling using the technique described in the previous section. The mean of the exponential distribution from which inter-arrival times are sampled is d'/n', where d' is the desired duration of $\mathcal{T}_c'$, and n' is the number of connection vectors in the resampling.

Using this technique, and under the assumption that $l = s/d$, the load offered by the resulting $\mathcal{T}_c'$ should be very close to the target load[5]. Figure 7.9 demonstrates that this is the case by comparing the offered loads of 1,000 simulated trace resamplings constructed using the technique in section 7.1.1 and another 1,000 resamplings using the byte-driven technique. The target load of the byte-driven resamplings was 161.89 Mbps, which was the average load in the original UNC 1 PM trace. The range of achieved offered loads is far narrower for the second technique, thanks to the variable number of connection vectors that are assigned to each $\mathcal{T}_c'$. The histogram in Figure 7.10 shows that the vast majority of the resamplings are very close to the target load ($\sigma = 0.41$ Mbps).

Figure 7.11 summarizes the results of 4 sets of 1,000 byte-drive Poisson resamplings. The plot uses the same type of visualization found in Figure 7.8. The error bars, barely visible in this case, illustrate the accurate load scaling that can be achieved with Byte-driven Poisson Resampling.

[5]Note that the duration of $\mathcal{T}_c'$ comes from random samples of an exponential distribution, so it can be slightly lower or higher that the intended d'. Given the light tail of the exponential distribution and the large number of samples, this deviation is necessarily very small.

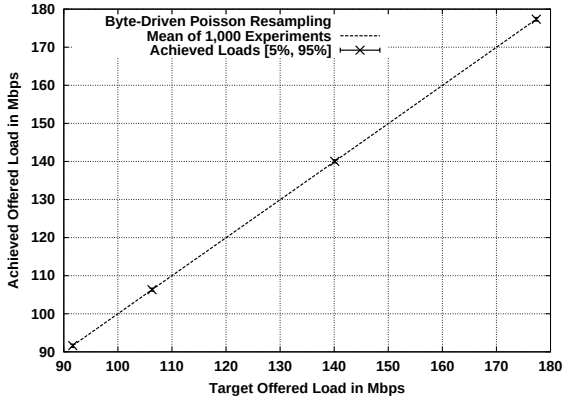

Figure 7.11: Analysis of the accuracy of byte-driven Poisson Resampling from 4,000 resamplings of UNC 1 PM (1,000 for each target offered load).

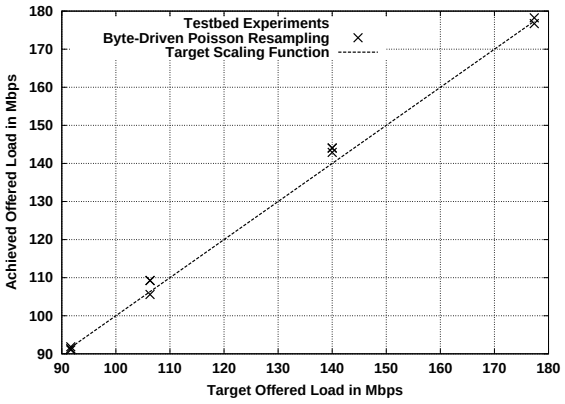

Figure 7.12: Analysis of the accuracy of byte-driven Poisson Resampling using source-level traces replay: replays of three separate resamplings of UNC 1 PM for each target offered load, illustrating the scaling down of load from the original 177.36 Mbps.

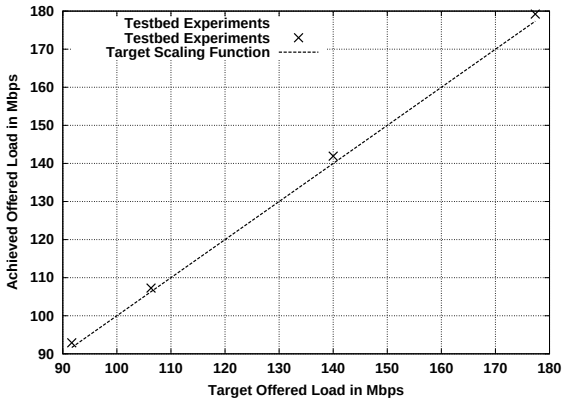

Figure 7.13: Analysis of the accuracy of byte-driven Poisson Resampling using testbed experiments: replay of one resampling of UNC 1 AM for each target offered load, illustrating the scaling up of load from the original 91.65 Mbps.

The previous analysis demonstrated that accurate load scaling requires control of the total number of bytes in $\mathcal{T}_c'$ rather than of the total number of connections. This demonstration was based on the computation of the offered load using equation 7.1. It is important to verify that the actual load generated during a testbed replay of $\mathcal{T}_c'$ is similar to the computed load. To show that this is indeed the case, we replayed a number of resampled traces with four different target loads. Each resampled trace was then constructed using byte-driven Poisson Resampling, with a duration of 1 hour. To eliminate startup and shutdown effects, we only considered the middle 40 minutes for computing the achieved load. Figure 7.12 summarizes the results of the experiments for the resamplings of the UNC 1 PM trace. Each point corresponds to a separate replay experiment, showing the target load on the x-axis and the achieved load on the y-axis. We ran three experiments for each target load, and the results show a good approximation of the intended scaling line. Several experiments achieved loads a few Megabytes above the target. In general, we expected the experiments to have slightly higher achieved loads, since the scaling method focuses on the offered payload, ignoring packetization overhead (*i.e.*, extra load from bytes in the packet headers). A more precise tuning of the offered load would take packetization into account, perhaps using a fixed constant to decrease the target payload, or by studying the total size (including headers) of each connection, as replayed in the same testbed environment. In any case, and given the above good results, it seems reasonable to ignore these further refinements.

The results in Figure 7.12 provide examples of *scaling down* the load of a trace, since the original load of UNC 1 PM was 177.36 Mbps, and 9 of the 12 experiments had target offered loads below this value. Scaling down the load of a trace using byte-driven resampling simply requires to choose a target load l' below the original load l, which in turns means that the s' of the resampling will be below the original s. These results confirm the close approximation of the target loads in the testbed experiments, where offered load is measured from real TCP segments (rather than computed using Equation 7.1). The plot shows for example that the three resamplings with target load 177.36 Mbps achieved loads of 176.72, 178.23 and 182.45 respectively. The impact of the TCP headers, retransmission and the slightly underestimated duration mentioned in the previous section is therefore very small. The results in Figure 7.13 provide an example of *scaling up* the load of a trace, since they correspond to byte-driven Poisson resamplings of UNC 1 AM, which had an original load of 91.65 Mbps. The resulting loads also approximate the intended targets very closely.

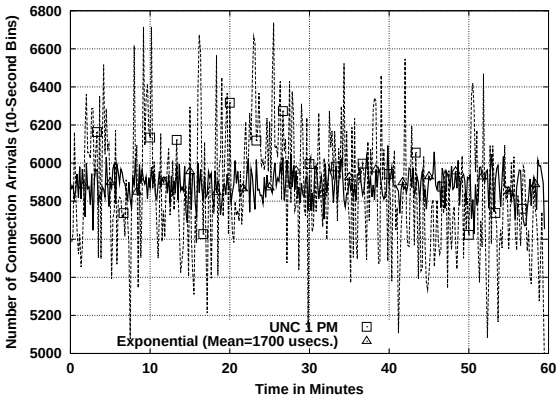

Figure 7.14: Connection arrival time series for UNC 1 PM (dashed line) and a Poisson arrival process with the same mean (solid line).

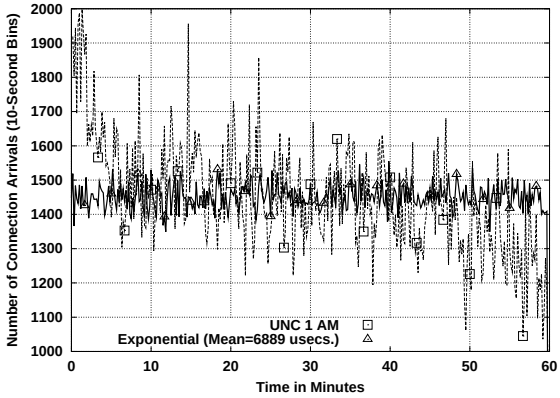

Figure 7.15: Connection arrival time series for UNC 1 AM and a Poisson arrivals process with the same mean.

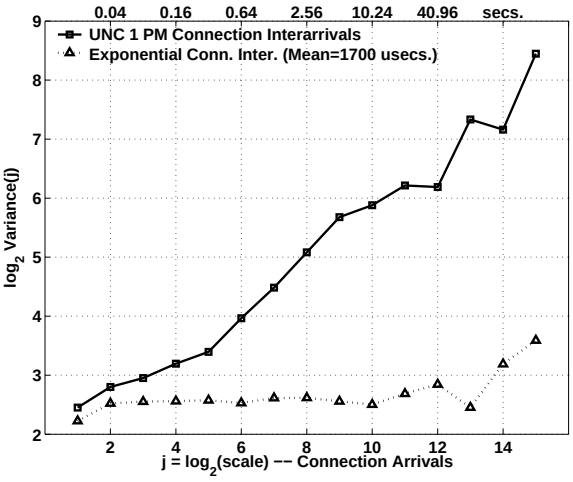

Figure 7.16: Wavelet spectra of the connection arrival time series for UNC 1 PM and a Poisson arrival process with the same mean.

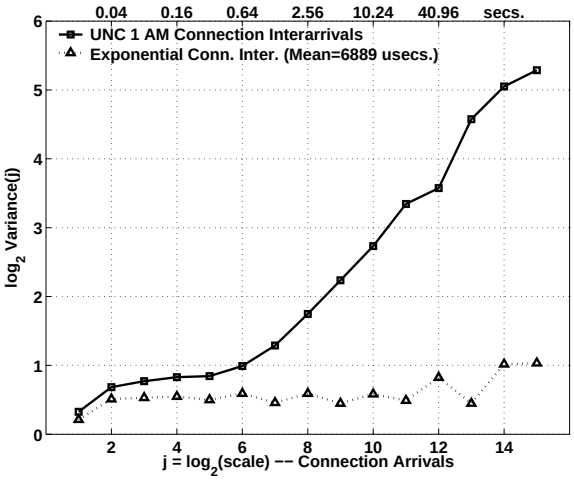

Figure 7.17: Wavelet spectra of the connection arrival time series for UNC 1 AM and a Poisson arrival process with the same mean.

Trace	Estimated Parameters
UNC 1 PM Conn. Arrivals	H=0.685041 C.I.=[0.646250, 0.723831]
Poisson $\mu = 1,698$ μsecs.	H=0.506069 C.I.=[0.467279, 0.544860]
UNC 1 AM Conn. Arrival	H=0.756533 C.I.=[0.717743, 0.795324]
Poisson $\mu = 6,889$ μsecs.	H=0.502217 C.I.=[0.463427, 0.541008]

Table 7.1: Estimated Hurst parameters and their confidence intervals for the connection arrival time series of UNC 1 PM and UNC 1 AM, and their Poisson arrival fits.

7.2 Block Resampling

The basic assumption of Poisson Resampling is that connection inter-arrivals are independent and identically distributed according to an exponential distribution, which results in a Poisson arrival process. While the choice of exponential inter-arrivals is reasonable given the measurement data presented in Figures 7.1 to 7.4, the arrival process may not necessarily be independent. On the one hand, we can argue that common application protocols make use of more than one connection, creating correlations among some start times. For example, web browsers often open several connections for downloading a web page. On the other hand, we focus on traces of highly aggregated traffic, where a large number of hosts start hundreds or even thousands of connections every second. The high aggregation could diminish or even eliminate completely any correlation structure in the connection arrival processes.

The analysis of our traces reveals non-negligible correlations in the connection arrival process. Figures 7.14 and 7.16 examine the arrival process for the UNC 1 PM trace. Using a time series of 10-millisecond bin counts, Figure 7.14 compares the burstiness of the original arrival process (dashed line) and that of a Poisson arrival process with the same mean inter-arrival time (solid line). The original arrival process was far more variable. Its standard deviation was equal to 346.07, while the one for the Poisson process was equal to 79.21. In order to further study the connection arrival process across a range of time-scales, we rely on wavelet analysis. Figure 7.16 shows the wavelet spectra of the original connection arrivals and a Poisson process with the same mean inter-arrival time. The Poisson process exhibits the expected flat spectrum of short-range dependent processes [HVA02]. On the contrary, the spectrum for the original connection arrivals follows a line with a substantial positive slope, which is characteristic of long-range dependent processes. The results of the wavelet-based estimation [AV98] of the Hurst parameters of these processes are given in table 7.1. The Poisson process has a Hurst parameter very close to the expected 0.5, while the original arrival process has a Hurst parameter of 0.685. This is consistent with moderate long-range dependence. For comparison, typical estimates of the Hurst parameter for packet and byte arrival processes are between 0.75 and 0.95, *i.e.*, typical packet and byte arrival processes exhibit significantly stronger long-range dependence than this connection arrival process.

We performed a similar analysis for the UNC 1 AM trace, and the results are shown in Figures 7.15 and 7.17. As in the previous case, the time series plot shows a connection arrival process that is significantly more bursty than that of a Poisson process with the same mean. Note however than in this case there is some degree of non-stationarity. We observe a significantly larger number of connections started in the first 5 minutes, and a significantly lower number started in the last 10 minutes. In this

case we compute the mean inter-arrival rate required to construct the Poisson arrivals using the middle 40 minutes of the trace. We therefore handle the effect of trace boundaries by ignoring the first and the last few minutes of the arrival process. The wavelet spectra for these middle 40 minutes and a Poisson process with the same mean arrival rate are shown in Figure 7.17. As in the UNC 1 PM case, the original connection arrival process exhibits clear long-range dependence. The estimated Hurst parameter in Table 7.1 reveals a somewhat stronger long-range dependence for the UNC 1 AM trace (0.757 *vs.* 0.685).

In summary, the connection arrival processes we have examined are consistent with significant long-range dependence. Therefore, it is desirable to develop the resampling and load scaling methods that can reproduce this structure, to cover experiments where the manner in which connections arrive is relevant for the network phenomenon studied using synthetic traffic. One example of this type of scenario is the evaluation of a router mechanism where the arrival of new connections creates new state in the router. For such a mechanism, a more bursty arrival process creates a more stringent workload, just like burstier traffic was shown by [BC98] to be more demanding on web server performance.

Poisson Resampling cannot reproduce this observed long-range dependence in the connection arrival process since its inter-arrivals times come from independently sampling an exponential distribution. For this reason, we propose a second resampling technique that can reproduce the long range dependence in the connection arrival process. The starting point is the intuition that dependencies between connection start times are far more likely to occur within relatively small periods. For example, web browsing results in new connections started according to the sequence of web page downloads and the way the browser opens new connections to the servers in which these pages are found. This results in brief bursts of connections whose start times are correlated. We use this intuition to develop a resampling method wherein the resampled objects are not individual connections, but groups of connections started during the same period, which we call *blocks*. The key idea of our *Block Resampling* method is that sampling blocks of connections rather than individual connections preserves the relative offsets of connection start times within blocks, and therefore the dependency structure[6] Our method is derived from the Moving Block Bootstrap method [ET93].

Block Resampling proceeds in the following manner: Given a trace $\mathcal{T}_c$, we divide it in blocks of duration β, so that the first block B_1 groups together connections started in the interval $[0, \beta)$, the second block B_2 groups together connections started in the interval $[\beta, 2\beta)$, and so on. The block resampled trace $\mathcal{T}_c'$ is obtained by concatenating randomly sampled blocks, and adjusting the start time

[6]We thank Peter Hall for suggesting the use of block bootstrapping in the context of the a-b-t model. The theoretical aspect of this idea are explored in [HNHC02], while this chapter focuses on its use to preserve the long-range dependence in connection arrivals and develops thinning and thickening methods to scale offered load in block-resampled traces.

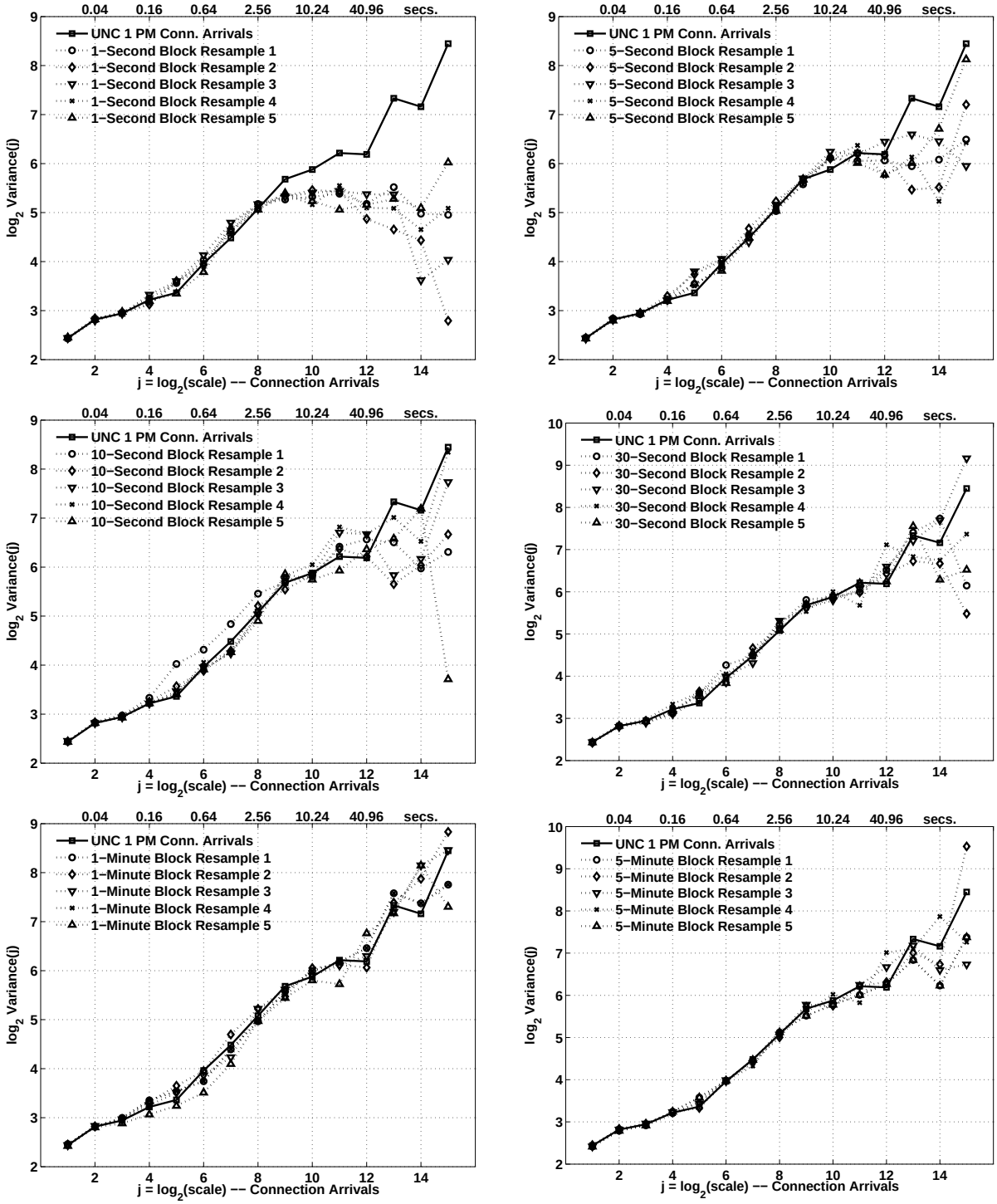

Figure 7.18: Block resamplings of UNC 1 PM: impact of different block lengths on the wavelet spectrum of the connection arrival time series.

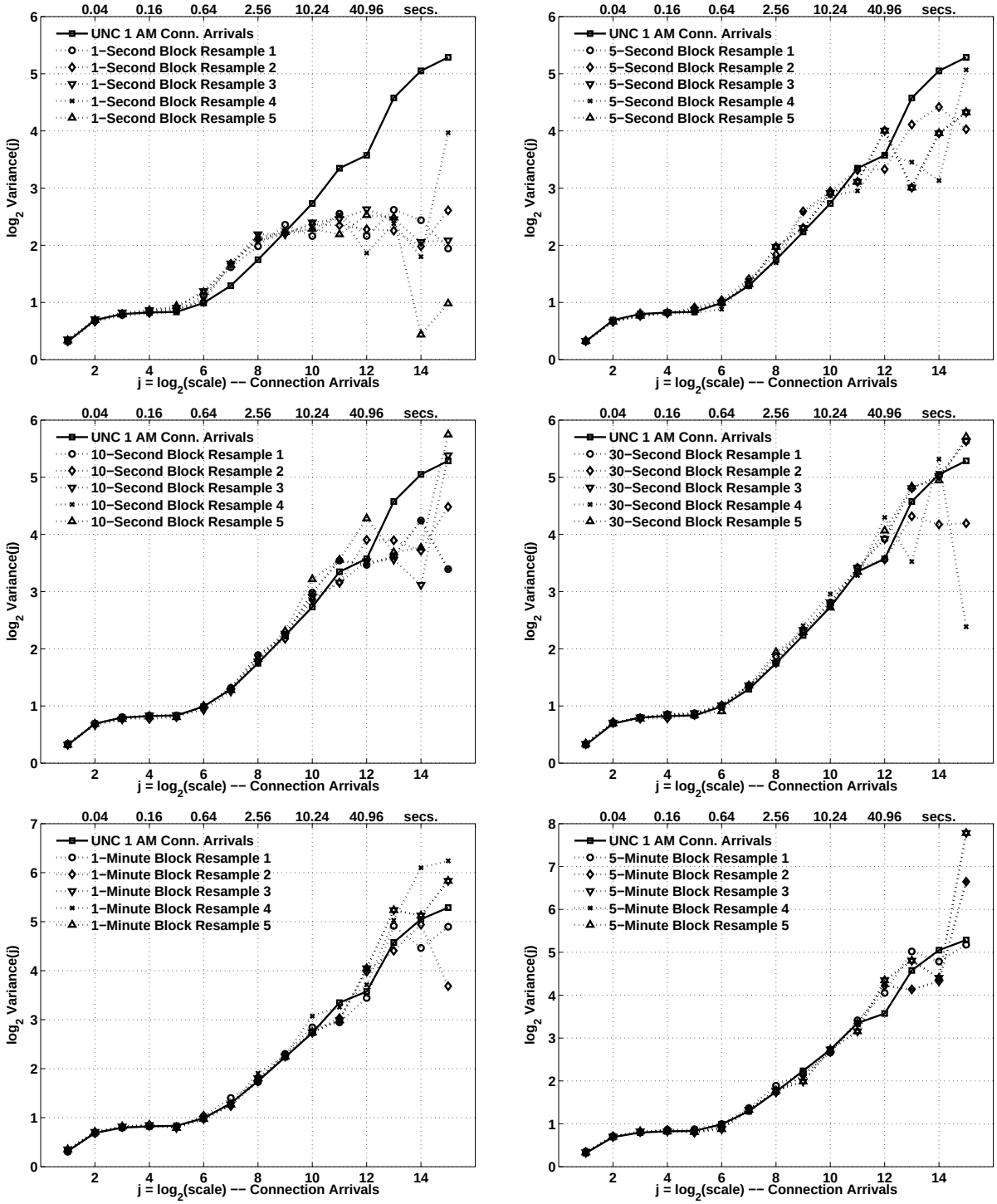

Figure 7.19: Block resamplings of UNC 1 AM: impact of different block lengths on the wavelet spectrum of the connection arrival time series.

of connections in each block by the time offset of the new location of this block. For example, if the random resampling puts block B_2 as the first block of $\mathcal{T}_c'$, the start times of the i-th connection vector in this block is set to $T_i - \beta$. Similarly, if B_2 is placed in the fourth location of $\mathcal{T}_c'$, the start times of the i-th connection in this block are set to $T_i + 2\beta$. More formally, when the j-th block B_j in the original trace becomes the k-th block B_k in the block resampling, the start time T_i of the i-th connection vector in B_j is set to

$$T_i' = T_i + (k - j)\beta.$$

Block Resampling chooses blocks for $\mathcal{T}_c'$ with replacement, making it possible to create new traces that are longer than the original $\mathcal{T}_c$ from which the blocks are obtained.

As pointed out by Efron and Tibshirani [ET93], choosing the block duration β can be a difficult problem. In our case, we found a clear trade-off between block duration and how well long-range dependence was preserved in the resampled trace. The shorter the block duration, the larger the number of distinct trace resamplings that can be performed from the same trace $\mathcal{T}_c$. This number is equal to $(d/\beta)!$ for resampled traces with the same duration d of the original trace. However, if the duration of the blocks is too small, the process of connection arrivals in the resampled trace exhibits a dependency structure that does not resemble the one in the original trace.

Figure 7.18 explores the impact of block duration on the long-range dependence of the connection arrival process in the resampled trace. The top left plot shows the wavelet spectra of the connection arrivals for UNC 1 PM and for 5 block resamplings where the block duration was 1 second. There is a clear and consistent flat region after octave 8, which shows that blocks of 1 second are too short to preserve the long-range dependence of the original connection arrival process. As the block duration is increased in subsequent plots, we observe an increasingly better match between the arrivals in the block resamplings and the arrivals in the original trace. Blocks with a duration of 30 seconds or 1 minute provide the best trade off between blocks that are large enough to ensure realistic long-range dependence in the connection arrival process, and blocks that are short enough to provide a large number of distinct resamplings. The same sensitivity analysis was performed for the UNC 1 AM trace and the results are shown in Figure 7.19. Block durations of 30 seconds or 1 minute are also shown to perform well.

As discussed earlier in this chapter, an important goal of trace resampling is the ability to preserve the target load of the original trace and to scale it up and down according to the needs of the experimenter. The analysis of a large set of Poisson resamplings revealed that offered load and number of connections are only loosely correlated. This motivated the use of a byte-driven version of Poisson Resampling

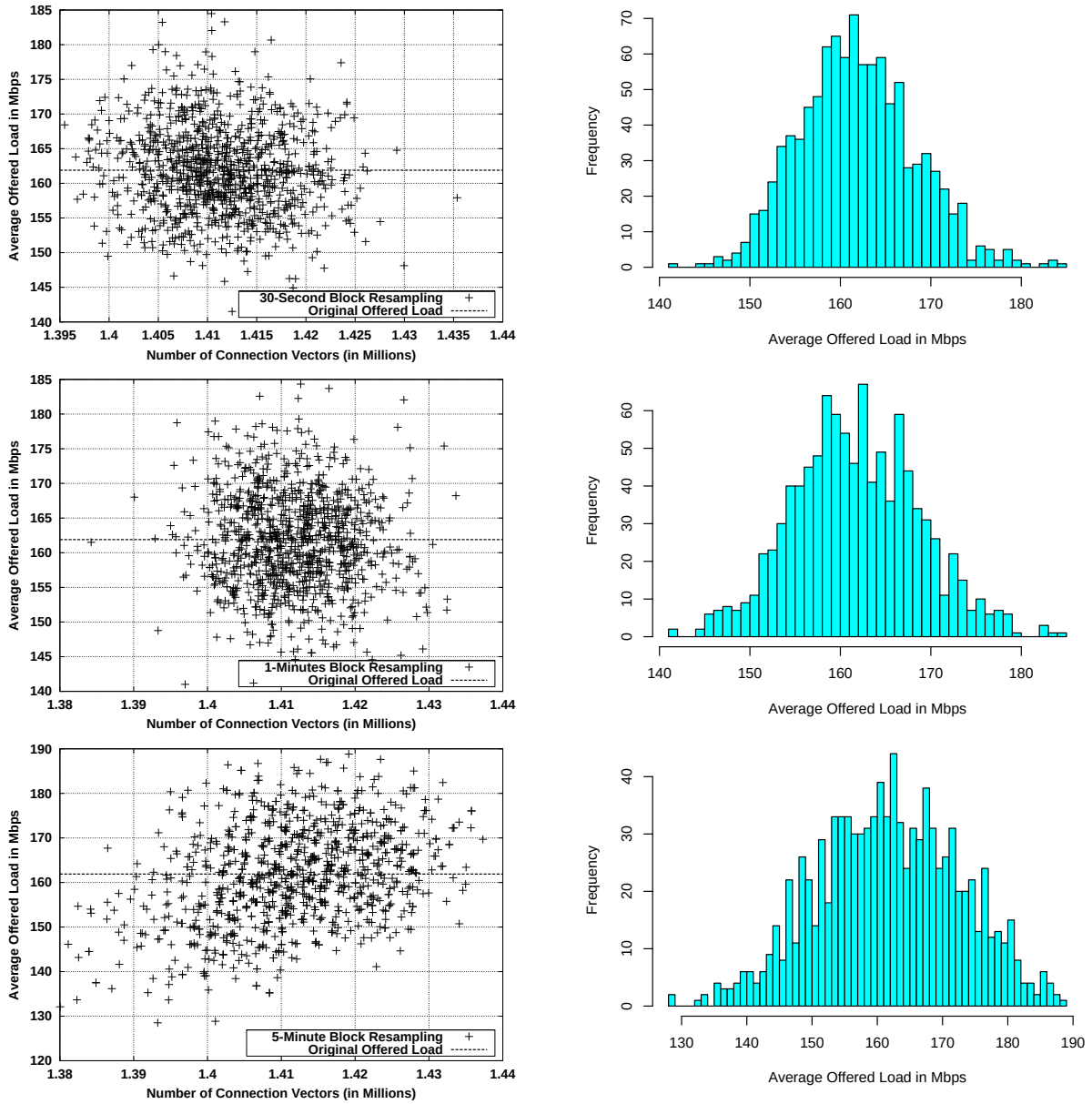

Figure 7.20: Block resamplings of UNC 1 PM: average offered load *vs.* number of connection vectors (left) and corresponding histograms of average offered loads (right) in 3,000 resamplings.

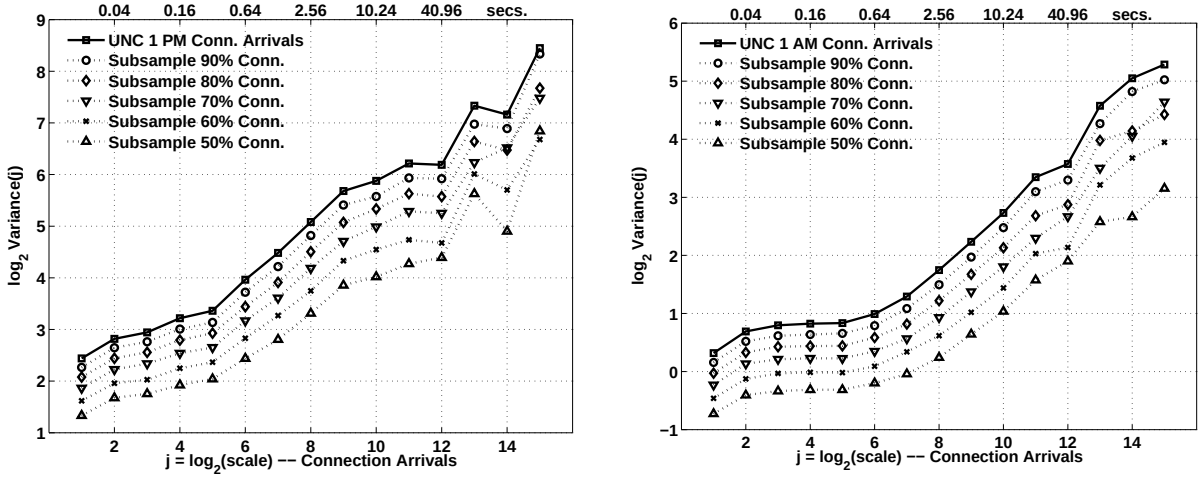

Figure 7.21: Wavelet spectra of several random subsamplings of the connection vectors in UNC 1 PM (left) and 1 AM (right)

which could achieve a very precise scaling of the load offered by the resampled trace. In the case of Block Resampling, the question is whether the averaging effect of grouping connections into blocks significantly diminishes the variability observed for the basic version of Poisson Resampling. We study this question by examining the offered load found in a large collection of block resampled traces. If the blocks had roughly uniform offered load, we would expect to generate similar offered load with each resampled trace. The results in Figure 7.20 do not confirm this expectation. The top row presents the analysis of 1,000 trace resamplings constructed by resampling UNC 1 PM using 30-second blocks. The average offered load was derived from the total payload computed using Equation 7.1. As shown in the scatterplot, the number of connections stayed within a narrow range, but the offered loads were far more variable. The histogram on the right further characterizes the distribution of offered loads in these trace resamplings. The use of blocks does not appear to have any benefit in terms of a more predictable load. This is not surprising given the known burstiness of the packet and byte arrival processes at many time-scales. If blocks were effective at smoothing out these processes, we would find little long-range dependence. This situation does not change for longer block durations, as shown in the middle and lower rows of Figure 7.20 for blocks of 1 and 5 minutes respectively. It is interesting to note the wider y-axis and range of the histogram for the 5-minutes blocks, which suggest even higher variability for this longer block duration.

The Block Resampling method described so far makes it possible to construct a resampled $\mathcal{T}_c'$ of arbitrary duration but it cannot be used to adjust its offered load. In order to perform this task, we can rely on *thinning*, when the offered load of $\mathcal{T}_c$ is above our intended offered load, and on *thickening*, when

Trace	Estimated Parameters
UNC 1 PM Conn. Arrivals	H=0.727540 C.I.=[0.701687, 0.753393]
Subsample 90% Conn.	H=0.724175 C.I.=[0.698322, 0.750028]
Subsample 80% Conn.	H=0.724046 C.I.=[0.698193, 0.749899]
Subsample 70% Conn.	H=0.718502 C.I.=[0.692649, 0.744354]
Subsample 60% Conn.	H=0.701378 C.I.=[0.675525, 0.727230]
Subsample 50% Conn.	H=0.701020 C.I.=[0.675167, 0.726872]
UNC 1 AM Conn. Arrivals	H=0.746591 C.I.=[0.720738, 0.772444]
Subsample 90% Conn.	H=0.738659 C.I.=[0.712806, 0.764512]
Subsample 80% Conn.	H=0.725030 C.I.=[0.699177, 0.750882]
Subsample 70% Conn.	H=0.715679 C.I.=[0.689827, 0.741532]
Subsample 60% Conn.	H=0.696723 C.I.=[0.670870, 0.722576]
Subsample 50% Conn.	H=0.691139 C.I.=[0.665287, 0.716992]

Table 7.2: Estimated Hurst parameters and their confidence intervals for five subsamplings obtained from the connection arrival time series of UNC 1 PM and UNC 1 AM

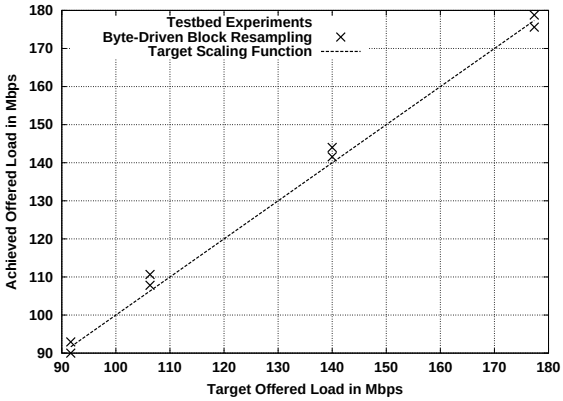

Figure 7.22: Analysis of the accuracy of byte-driven Block Resampling using source-level trace replay: replays of two separate resamplings of UNC 1 PM for each target offered load, illustrating the scaling down of load from the original 177.36 Mbps.

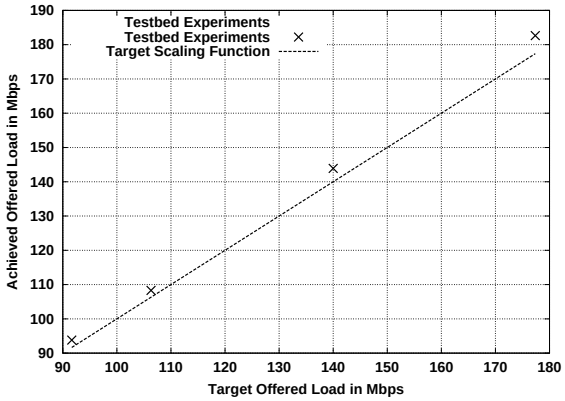

Figure 7.23: Analysis of the accuracy of byte-driven Block Resampling using source-level trace replay: replay of one resampling of UNC 1 AM for each target offered load, illustrating the scaling up of load from the original 91.65 Mbps.

the offered load of $\mathcal{T}_c$ is below our intended offered load. Block thinning involves randomly removing connections from $\mathcal{T}_c'$. Theoretical work by Hohn and Veitch [HV03]has shown that the thinning of a long-range dependent process does not change its long-range dependence structure. Our own experimentation confirms this result. Figure 7.21 shows the wavelet spectra of thinned versions of the connection arrivals in the UNC 1 PM trace (left) and the UNC 1 AM trace (right). The overall energy level decreases as the fraction of connections removed from each block increases. However, the spectra maintain their shapes, which demonstrates that the degree of the long-range dependence remains unchanged. The estimated Hurst parameters for these two traces is presented in Table 7.2. The values reveal only a moderate decrease in the Hurst parameter even when half of the connections are dropped.

Block thickening consists of combining more than one block in each of the disjoint intervals of $\mathcal{T}_c'$, *i.e.*, to "fusion" one or more blocks from $\mathcal{T}_c$ to form a single block in $\mathcal{T}_c'$. This makes the offered load a multiple of the original load. For example, to double the load, the connection vectors of two randomly chosen blocks will be placed in the first interval, those from another pair of randomly chosen blocks will be placed in the second interval, and so on. The new start times of the connection vectors in the resampled trace are computed using Equation 7.2, but being careful to use the right j for each connection vector.

To achieve offered loads that are not a multiple of the original load, we can combine basic thickening and thinning using a two-step process. The first step is to create a preliminary version of $\mathcal{T}_c'$ by combining as many blocks as possible without exceeding the target load. The second step is to "complete" this trace by combining it with another block-resampled trace that has been thinned in such a manner that the combined load of the two resampled traces matches the intended load. For example, in order to create a $\mathcal{T}_c'$ with 2.5 times the load of $\mathcal{T}_c$, a first thickened trace $\mathcal{T}_c^{tk}$ is created by combining two blocks in each position. This trace is then combined with second trace $\mathcal{T}_c^{tn}$ that has been thinned to half of the offered load of $\mathcal{T}_c$. From our analysis in Figure 7.20, we can see that $\mathcal{T}_c^{tk}$ is not necessarily equal to twice the offered load of $\mathcal{T}_c$. For this reason $\mathcal{T}_c^{tn}$ is actually thinned to exactly the offered load needed to complement $\mathcal{T}_c^{tk}$, and not just to half of the original offered load This careful thinning makes the scaling match the intended load in a highly precise manner. We can therefore achieve any intended load with the Block Resampling method, so it is as flexible as Poisson Resampling. In accordance with our earlier analysis, accurate thinning cannot rely on any correlation between the number of connections and the offered load, so it must be driven by Equation 7.1, just like byte-driven Poisson Resampling. Therefore, our final resampling technique is Byte-driven Block Resampling.

Figures 7.22 and 7.23 show the result of several testbed experiments where Byte-driven Block Resampling is used to create new traces. The results demonstrate that traces resampled using this method achieve a very good approximation of the target offered loads. As in the case of Byte-driven Poisson Resampling, the achieved loads are slightly higher than target ones due to the packetization overhead, which is not taken into account in the resampling.

One interesting question is whether the effort to preserve the scaling of the connection arrival process has any effect on the generated traffic aggregate. To understand this question, we can compare the process of packet (or byte) arrivals from block resamplings and from Poisson Resampling, since the former fully preserves connection arrival long-range dependence and the latter fully eliminates it. Figure

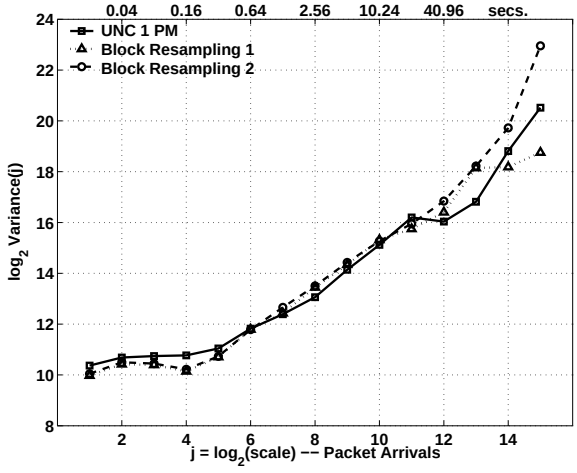

Figure 7.24: Wavelet spectra of the packet arrival time series for UNC 1 PM and the source-level trace replays of two block resamplings of this trace.

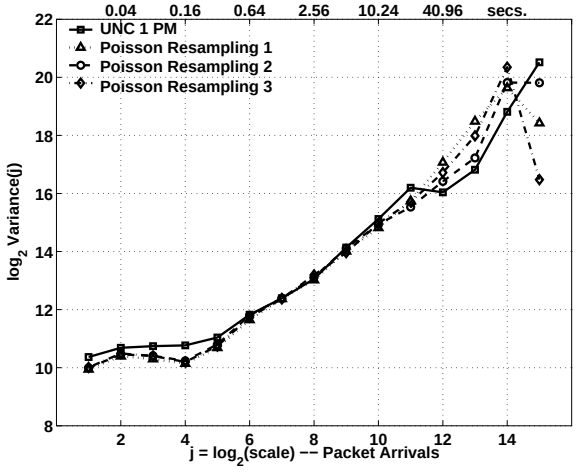

Figure 7.25: Wavelet spectra of the packet arrival time series for UNC 1 PM and the source-level trace replays of three Poisson resamplings of this trace.

7.24 shows the wavelet spectra of the packet arrivals in UNC 1 PM and those in two testbed experiments where byte-driven block resamplings of UNC 1 PM were replayed. Figure 7.25 shows the same wavelet spectrum of the packet arrivals in UNC 1 PM, and also the spectra from three testbed experiments where byte-driven Poisson resamplings of UNC 1 PM were replayed. Both resampling methods achieve equally good approximations of the packet scaling found in the original trace. In other words, according to this type of analysis, the simpler Poisson Resampling method performs as well as the more elaborate Block Resampling method. This is a confirmation, using a closed-loop traffic generation approach, of the results by Hohn *et al.* in [HVA02], which were obtained using (open-loop) semi-experiments. This is not to say that long-range dependence in the arrival of connections (*e.g.*, arrival of flow state or cache misses to a router) can be safely ignored, since other metrics and experimental results may be more sensitive to this characteristic of the synthetic traffic.

7.3 Summary

Our basic traffic generation method, source-level trace replay, results in highly realistic synthetic traffic. This method is however inflexible, in the sense that the same connection vectors are started at the same relative times in every replay. In this chapter, we proposed two methods for resampling an original trace of connection vectors, to create a new trace with similar statistical characteristics. This similarity is defined in terms of source-level behavior and network-level parameters, so the resampling methods

also modify connection vector start times. Our first resampling method is Poisson Resampling, which chooses connections vectors at random and assigns them exponentially distributed inter-arrival times. Our measurement results demonstrated that this choice of the inter-arrival distribution is appropriate, in the sense that the marginal distribution of the connection inter-arrival in every trace we examined is remarkably consistent with the exponential distribution. Our second resampling method is Block Resampling, which chooses blocks of connection vectors at random. Unlike Poisson Resampling, Block Resampling preserves the dependency structure of the original connection arrival process. This makes it possible to reproduce the moderate long-range dependence that we observe in the connection arrivals of our traces.

Besides presenting two resampling methods, we also studied how to control the offered load by a resampled trace. Firstly, we demonstrated that the number of connections and the average offered load are not strongly correlated. This means that controlling the number of connections in the resamplings does not provide a good way of creating resampled traces with a specific target offered load. This is a common requirement when a set of experiments covers a range of offered loads in an empirical study. In order to address this difficulty, we propose to drive the resampling by a target total size of the ADUs in the resampling rather than by a target number of connections. We used this approach to develop byte-driven versions of Poisson Resampling and Block Resampling, which are shown to result in highly predictable offered loads.