Traffic Load Scaling for Network Design

K. Sleurs*, J. Potemans, B. Van den Broeck, J. Theunis, D. Li, E. Van Lil, A. Van de Capelle
Department of Electrical Engineering – ESAT, TELEMIC division
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
kristof.sleurs@esat.kuleuven.be

Keywords: Traffic scaling, Network design, Self-similarity, Load scaling, Performance study.

Abstract

While designing network applications, one has to check their performance on a network that is loaded by realistic background traffic. For this purpose, a packet stream can be captured on a network to obtain real background traffic. When the application has to be tested under various network loads, one can use straightforward techniques to alter the original captured traffic trace; for example faster replay of the packet stream. The problem with these simple techniques is that the packet stream will be altered in many ways, and thus a simple multiplication of the load inflicted by the packet stream on a network cannot be guaranteed. In this paper we will first describe a few simple techniques. Then, more complex techniques that better approach the goal of multiplying the traffic load by a known factor will be presented.

1.1. INTRODUCTION

Every day work is done on designing software for new internet applications. All the internet applications available today are supported by hardware that is continually optimized. Before integrating these products in real-life networks, some testing has to be done on their performance. For example when a networking application that implements some functionality between two peers has to be tested, the communication line between these two peers should be loaded with traffic comparable to real internet traffic. Other examples are hardware networking products like routers and switches. Their performance should be tested under traffic loads that at least resemble real internet traffic. Therefore, companies need traffic streams to either use as background traffic for their networking applications or as input to their hardware products.
One possibility is to capture this traffic on a reference network [1]. The drawback of this technique is that when the applications have to be tested under different network loads, for each load another packet stream with the right number of packets per second has to be located and captured from a real-life network. More problems occur due to the fractal behavior of internet traffic. To make a good comparison, all the different streams must have a comparable self-similar behavior if the sole effect of increasing load is to be tested. This makes this technique quite difficult to use in practice.
Other methods to change the load inflicted by a packet stream on a network are thus needed. In this paper we describe some methods to set this load to a desirable level, using only one packet stream captured from a reference network. We present a technique that takes as little computational effort as possible, while creating a real multiplication of the load inflicted by the packet stream.
Important to note is that for this study, the only data necessary from the reference network are the timestamps of the packets arriving at one point. A sequence of timestamps is then generated with the same characteristics as the reference stream, but with a different load. This packet stream could then be used, for example, to load a backbone connection between two routers. We thus will not model user events or servers, but remain on the lowest levels of the OSI-model.
First some terms like self-similarity will be looked at. Then the goals we want to achieve are described more in detail. This is followed by a description and study of some simple, straightforward techniques to double the load of a packet stream. The results of these simple methods however are found to be unsatisfying. More advanced methods such as rotation of the bin count vector and making use of a traffic model are studied. Finally, some remarks are made on the discrete scaling of traffic with larger factors. Scaling with factors less than one is also shortly touched.

* Kristof Sleurs is Research Assistant of the Fund for Scientific Research Flanders (FWO-Vlaanderen), Belgium.

2. PRELIMINARIES

2.1. Discrete Traffic Models

Discrete traffic models divide packet streams into time intervals. A certain length is chosen for this time interval and all timestamps within a certain time interval are placed in one 'bin'. Like this, a bin vector is obtained with each bin representing the number of packets that arrived in that time interval. The bin vector can be represented by a timeseries variable *X*, with X_k representing the number of packets in bin *k*.

Another important concept is aggregation. An aggregated time series $X^{(m)}$ with aggregation level *m* is obtained by averaging the original time series *X* over non-overlapping blocks of *m* intervals:

$$X_j^{(m)} = \frac{1}{m} \sum_{k=(j-1)m+1}^{jm} X_k$$

with $j = 1, 2, ..., n/m$, where *n* represents the number of elements in the original time series.

This principle can be applied to the bin count vector. By aggregating, a packet stream is in fact averaged out over a certain time interval, normally reducing its peaked behavior.

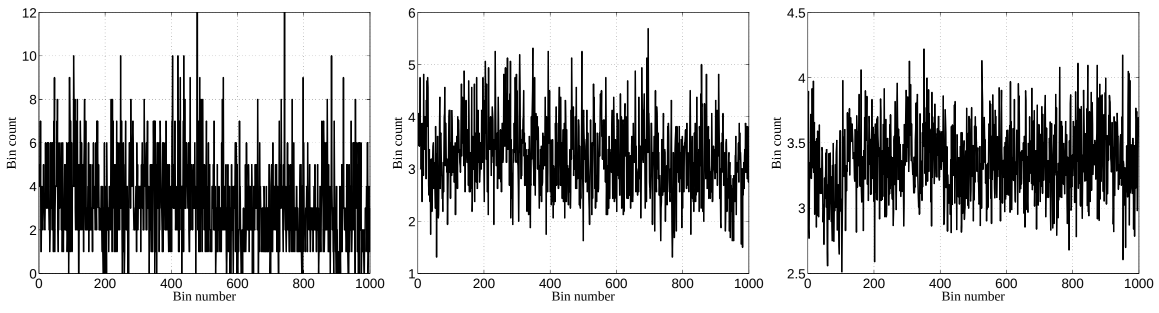


Figure 1. Bin count vector at different aggregation levels.
(a) Bin count vector, not aggregated, (b) Bin count vector aggregated with m=2^4, (c) Bin count vector aggregated with m=2^8.

2.2. Used Traffic Traces

The traffic trace used for this study is part of the AbileneIII trace. This is a publicly available OC192c backbone trace. It was collected on the Abilene network at the link between the Indianapolis router node and Kansas City on June 1st, 2004. The '20040601-200000-1.gz' part of this trace was used. The length of this packet stream is 600 seconds, with an average number of packets per second of 95428.
This data set is available online thanks to the NLANR Network Analysis Infrastructure [2].

2.3. Fractal Behavior of Network Traffic

When we look at the bin count vector of the AbileneIII trace, it seems to behave in a very bursty way. Instinctively one would think that when aggregated, the peaks will average out and the bin vector will be less bursty. In Figure 1 it is shown that this doesn't happen for the captured traffic trace. In fact, the aggregated and the original bin vector are very much alike. This is why this behavior is called self-similarity [3-6].

The reason for this odd behavior is the correlation structure of network traffic. To show this, the original bin count vector is randomized and then again aggregated like before. Figure 3 shows that the bursty behavior disappears with increasing aggregation level. The correlation structure in real, self-similar traffic has the characteristic of not changing with aggregation.

Another way to look at this self-similarity is through a variance-time plot. This draws the variance of a bin vector against the aggregation level. Now, for the memoryless Poisson generation process (which implies exponentially distributed interarrival times), the logarithmic plot should decrease linearly with slope -1. In Figure 2 it can be seen that for the AbileneIII trace this indeed is not the case, what again is a clear proof of the non-Poisson behavior of real traffic [7-8].

The implications of this fractal behavior are quite important. The consequence is that in reality, when more users generate traffic – the network is more loaded – traffic will not average out, but will become even more bursty. This leads to possible congestion in routers and could lead to increased packet loss. This makes it important to take the fractal behavior of real traffic into account when testing a network application or network hardware [9].

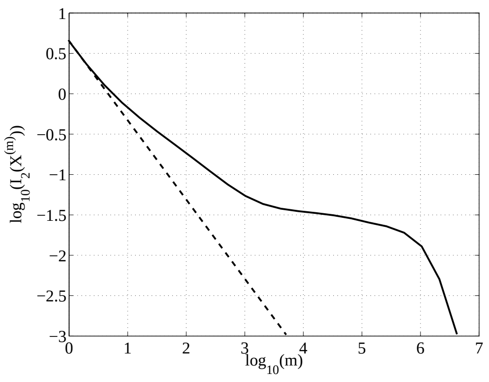


Figure 2. Variance-time plot for the AbileneIII traffic trace.

3. GOAL

The goal we want to achieve is generating a packet stream which scales the load imposed on the network under study with a certain factor compared to the original, measured reference trace. By scaling we mean multiplying the average number of packets per second with a factor, but also multiplying the moment values for all aggregation levels with this factor. Like this, the original moment behavior will be retained after scaling.

A possible way to achieve this goal is modeling the measured traffic, altering this modeled traffic in some way, and then generating a generic packet stream with the desired characteristics. Several papers are available on this method [10-11]. The drawback of these techniques is their complexity. In this paper we will use one of these models to compare to other techniques of scaling the load.

We will first examine some straightforward techniques and apply these to the problem of doubling the traffic load. Related work is done in [12]. Eventually we want to develop a technique that works as well as modeling the packet stream, but with less complexity and less computational effort. The strategy we follow throughout this paper is creating – next to the original packet stream – a second stream, independent of the first, and with the same length, number of packets and moment behavior. Next, these streams will be added together to result in a double load packet stream.

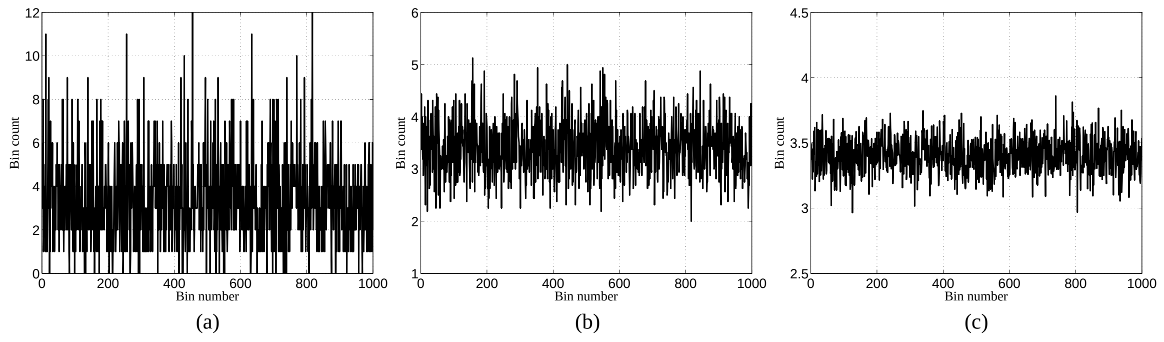


Figure 3. Randomized bin count vector at different aggregation levels.
(a) Bin count vector, not aggregated, (b) Bin count vector aggregated with m=2^4, (c) Bin count vector aggregated with m=2^8.

To study the self-similarity characteristics, the packet streams will be divided into bins. The number of bins was chosen according to the smallest time scale at which correlation is present. In case of the AbileneIII stream, this meant a division into 2^{24} bins. This equals a bin length of 35.76 µs. For computational reasons, we always aggregate by powers of two, and therefore the packet streams under study have to be truncated on a number of bins equal to a power of two. When we choose a whole power of two as the number of bins, no packets will be lost.
Additive centred moments are used. In case of the second order moment $I_2(X)$, this comes down to the mean squared deviation (also called variance):

$$I_2(X) = Var(X) = E\left[(X - E[X])^2\right].$$

The unnormalized centred third order moment $I_3(X)$, which measures how symmetric a distribution is, is defined as follows:

$$I_3(X) = E\left[(X - E[X])^3\right].$$

The additive centred fourth order moment $I_4(X)$, measuring the weight of a distribution's tail, is described by:

$$I_4(X) = E\left[(X - E[X])^4\right] - 3(I_2(X))^2 .$$

Skewness and kurtosis were not used because they lack the additive property. We study second, third and fourth order moments at different aggregation levels to characterize the distribution of bin counts as precise as possible.

When the second packet stream is generated, bin counts of both streams will be added up. Because the two streams ideally have the same moment behavior and are independent of each other, their moments will simply add up (thanks to the moments' additive property), and a new stream will be generated with moments equaling twice the moments of the original stream at each aggregation level.
To make a good comparison of the queuing behavior between different techniques based on their moment behavior, the timestamps of packets within a bin are randomized. Like this, differences between techniques will be due to the difference in bin count distribution on certain aggregation levels and not to intra-bin timestamp distributions.

4. SIMPLE TECHNIQUES

4.1. Trace Division

First, individual streams were extracted from the measured trace, based on the 4-tuple <Source IP address, Source port, Destination IP address, Destination port>. Then these streams were randomly grouped into two large packet streams. Because of the large number of streams, eventually both parts contain a comparable number of packets. Part 1 will further be taken as the reference packet stream, and designated as the original packet stream. Part 2 is in fact an approximation of the packet stream that we are searching for to append to the original packet stream. This first technique is used mainly for comparison purposes, because an original packet stream of course can't be doubled like this.

4.2. Bin Count Doubling

The second quite straightforward technique creates a second packet stream by taking the same bin count vector as the original packet stream. Packets inside the bins are then again randomized. Like this a second packet stream is obtained. These timestamps are appended to the original timestamps. This way, two packet streams with the same length, number of packets and moment behavior are added. This technique can also be seen as simply doubling the bin counts of the original packet stream. One major drawback of this 'bin count doubling' is the fact that both streams are certainly not independent. This is clearly visible when the mean and variance are calculated:

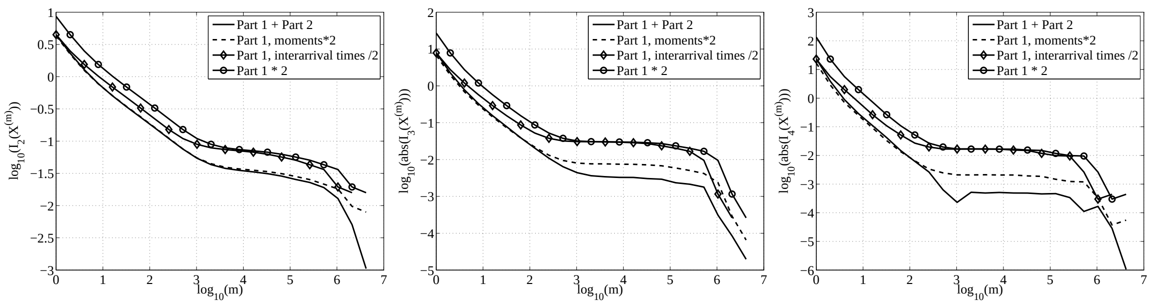
$$\overline{Y} = E[Y] = \frac{1}{n}\sum_k Y_k = \frac{2}{n}\sum_k X_k = 2 \cdot \overline{X}$$

$$\begin{aligned}\sigma_y^2 &= E[Y^2] - E^2[Y] \\ &= \frac{4}{n}\sum_k X_k^2 - \frac{4}{n^2}\left(\sum_k X_k\right)^2 . \\ &= 2^2 \cdot \sigma_x^2\end{aligned}$$

Here the original bin count is represented by *X*, and the doubled packet stream is designated by *Y*. Both streams are divided into *n* bins, and X_k and Y_k represent the number of packets in bin *k* for both streams. The mean number of packets per bin indeed doubles, but the variance gets multiplied by four instead of two. The same happens to the higher order moments; the third and fourth order moment get multiplied by respectively eight and sixteen. This effect is, as already stated, due to the high correlation between the two added packet streams.

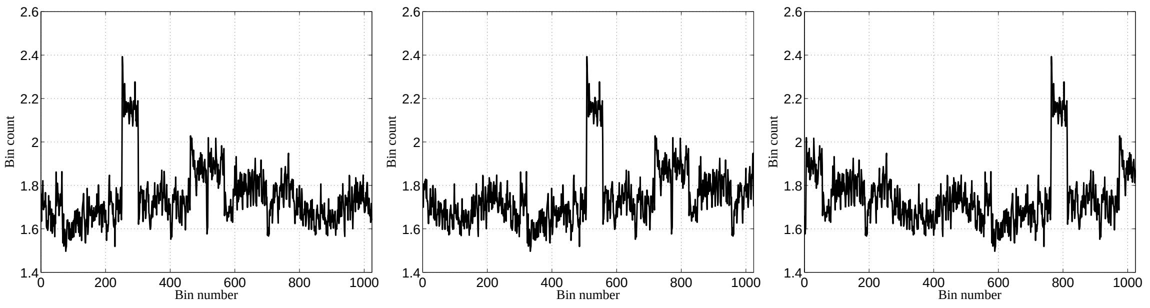
4.3. Halving Interarrival Times

Thirdly, the interarrival times of the original packet stream are halved. This technique doesn't really create a second stream, but alters the original stream. In this case, it's more difficult to predict the effect on the moments of the resulting stream, but a few calculations will give some insight:



(a) (b) (c)

Figure 4. Moment plots for the simple scenarios.
(a) Second order moment, (b) Third order moment, (c) Fourth order moment.



(a) (b) (c)

Figure 5. Rotations of the bin count vector of AbileneIII, Part 1.
(a) Original bin count vector, (b) Rotated bin count vector, offset a quarter of the vector length, (c) Rotated bin count vector, offset half the vector length.

$$\overline{Z} = \frac{1}{n/2}\sum_{j} Z_j = \frac{2}{n}\sum_{j=1}^{n/2}(X_{2j-1}+X_{2j})$$
$$= \frac{2}{n}\sum_{k=1}^{n} X_k = 2\cdot\overline{X}$$

$$\sigma_z^2 = \frac{1}{n/2}\sum_{j=1}^{n/2} Z_j^2 - \frac{4}{n^2}\left(\sum_{j=1}^{n/2} Z_j\right)^2$$
$$= \frac{2}{n}\sum_{k} X_k^2 - \frac{4}{n^2}\left(\sum_{k} X_k\right)^2 + \frac{4}{n}\sum_{j=1}^{n/2} X_{2j}X_{2j-1} \quad .$$

Here the original bin count is again represented by *X*, and the half interarrival times packet stream is designated by *Z*. The relationship between the original and the new time series is given by: $Z_j = X_{2j-1} + X_{2j}$. The halving of interarrival times is equal to adding the packets in each two subsequent bins together. This principle is visible in the previous formula.
We define autocovariance as follows:

$$R(k) = E[X_i X_{i+k}] \ .$$

We use this autocovariance with *k=0* and *k=1*, and define the altered autocovariance:

$$R(0) = \frac{1}{n}\sum_{k} X_k^2$$
$$R(1) = \frac{1}{n}\sum_{k} X_k X_{k+1}$$
$$R^*(1) = \frac{2}{n}\sum_{k=odd} X_k X_{k+1} \ .$$

The resulting variance can be written as:

$$\sigma_z^2 = 2^2\cdot\sigma_x^2 - 2\cdot R(0) + 2\cdot R^*(1) \ .$$

When a stationary packet stream is considered, *R(1)* and *R*(1)* are equal if enough bins are taken into account. With increasing aggregation level, *R(0)* and *R(1)* get more equal, hence halving the interarrival times gets identical to doubling the bin count for large interarrival times.
For higher order moments, calculations get a lot more cumbersome. From Figure 4, the higher order moment behavior can be regarded as very comparable to second order moment behavior.

To conclude, second, third and fourth order moments are plotted in Figure 4. As it can be seen, the deviation from the ideal scenario (Part 1, moments*2) is quite large, especially for third and fourth order moments. Even for the artificially generated double load stream (Part 1 + Part 2), significant deviations are visible for higher order moments. Another remark is that indeed doubling bin counts and halving interarrival times have the same effect on the moments for large aggregation levels. Note that the differences for the highest aggregation levels are due to the small number of remaining bins and the resulting non-convergence of the moments.

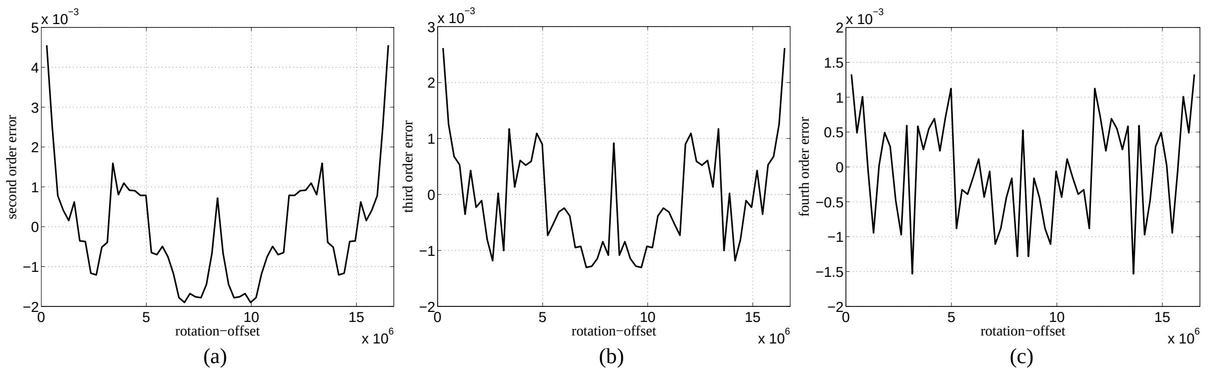


Figure 6. Deviation of the moments of the composed packet streams due to correlation between original and rotated bin count vectors. Errors are plotted relative to the error for zero rotation.
(a) Relative second order moment deviation, (b) Relative third order moment deviation, (c) Relative fourth order moment deviation.

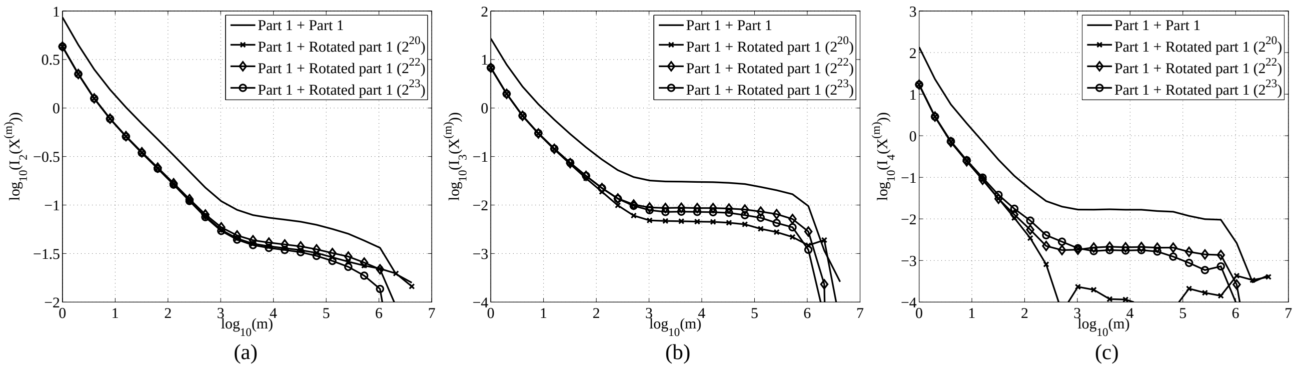


Figure 7. Moments for the sum of the original bin vector and a rotated bin vector (rotation offset is shown in legend).
(a) Second order moment, (b) Third order moment, (c) Fourth order moment.

5. ADVANCED TECHNIQUES

5.1. Bin Vector Rotation

After these fairly straightforward techniques, we try another bin count altering technique to obtain better results. The main problem with bin count doubling was the large correlation between the two - basically equal - bin count vectors. Next, the second bin count vector will be rotated before summing it to the original bin vector. Like this, the optimal rotation offset can be found to get the correlation with the original bin vector as small as possible.

First, the rotation principle is shown in Figure 5. Next to the original bin count vector (Figure 5a), the rotated bin vector is shown, respectively with a quarter length (Figure 5b) and half length (Figure 5c) rotation offset. The rotation offset is the number of bins that are taken from the end of the bin count vector and are placed at the start of the vector.
The moment behavior of the original bin count vector will not change below aggregation level 2^{n+1} when rotated with an offset of 2^n. For higher aggregation levels, our empirical study has shown the deviation to remain very small. To attain an ideal doubling of the packet stream, the rotation offset has to be found that generates the smallest total deviation for the second, third and fourth order moments due to correlation, taking into account every aggregation level. The relative value of the correlation error compared to the error for zero rotation is plotted in Figure 6 for aggregation level 0. Obviously, this correlation error is symmetric about half the length of the packet stream. This can be quite easily understood when looking at the characteristics of autocorrelation. Another remark that has to be made is the heavily fluctuating behavior of these errors, especially for higher order moments.

When the rotation technique is used, the problem remains of choosing the optimal rotation offset. When the rotation offset is chosen outside the initial decrease of the error shown in Figure 6, the deviation for second, third and fourth order moments does not get unacceptably high. The relative change in deviation is largest for the second order moment. For optimal results, moment errors at all aggregation levels have to be considered. However, Figure 7 shows that the random choice of an offset outside the initial decrease yields results within acceptable ranges, at least for second and third order moments.

When an optimal solution is desired though, we try to find the rotation offset that results in the least total relative error. Searching through all possible rotation offsets would be too time-consuming. Therefore, first the change in rotation offset that changes error significantly is sought. Based on this study, we search for the smallest error for rotation offsets from 0 to half the packet stream's length (2^{24}) in steps of 2^{20}. This results in a rotation offset of 2^{21} + 2^{20} for the AbileneIII trace. To achieve the best results, rotation offsets in the neighborhood of this value can be searched for small improvement of errors. This isn't done here because the gain will be small compared to the extra computation time necessary. For the rotated bin count vector, random intra-bin timestamps are calculated and appended to the original packet stream. Queuing behavior will be looked at later.

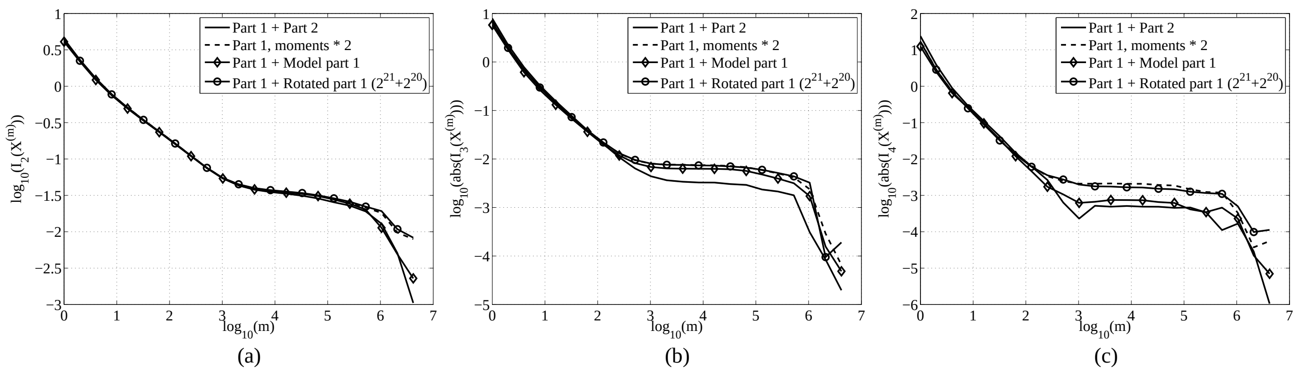


Figure 8. Plots of the moments for the advanced scenarios.
(a) Second order moment, (b) Third order moment, (c) Fourth order moment.

5.2. Traffic Model Packet Generation

Another possibility to create a second packet stream, independent of the original stream and with comparable moment behavior, is by using traffic models. The model used here is an additive, dyadic lognormal model [10], whereto the moments of the original packet stream are inputted, and a second packet stream with the same length is generated. The model fits the second and third order moments of the packet stream.

The resulting moment behavior of both techniques is shown in Figure 8. The improvement compared to the simple techniques is clearly noticable. For comparison, the first simple technique (summing the two parts of the trace, not applicable in reality) is also drawn. The rotation of the bin count vector yields the best results, closely followed by the use of a model-based generated packet stream. The deviation of the model-based generated packet stream is especially larger for the fourth order moment. This is due to the fourth order moment not being fitted by the model. If the computational effort is also taken into account, the rotation of the bin count vector can be considered as the best choice.

6. QUEUING BEHAVIOR

To study the practical implications of these different techniques and their moments behavior, queuing characteristics such as delay distribution and drop probability are simulated. These simulations were conducted in Matlab with a modeled single queue system with a constant packet service time of 7.675 µs for the AbileneIII packet stream. This corresponds to a load of 75 %.
The plots in Figure 10 confirm that bin vector rotation and packet stream modeling yield far better results than the simple techniques. When we look at delay distribution for example, the former techniques predict a probability far less than 10^{-6} for delays larger than 10 milliseconds. The latter techniques predict a much larger probability of almost 0.5 percent for this magnitude of delays. In case of average drop probabilities, the simple techniques predict a needed system capacity twice to three times as large as predicted by the more accurate techniques for average drop probabilities of about 1 percent.

There still are differences between the rotation technique and the modeled packet stream though. A possible explanation could be the better modeling of the fourth order moment by the rotation technique. This could be resolved by also fitting the fourth order moments by the traffic model.

7. SCALING WITH LARGER FACTORS

Until now, only scaling with a factor 2 was considered. When we want to scale with larger factors, the same rotation principle can be used. The maximum scaling factor to keep the correlation error under control can be predicted by looking at Figure 6. The fluctuations of the second order moment deviation are the largest, and are therefore shown in more detail in Figure 11.

When we plot the maximum absolute value of the error for all rotation offsets larger than a given value, Figure 11 shows that for the second order moment a rotation offset greater than 6 x 10^5 keeps the error smaller than 0.32 % of the error without rotation. Relative third and fourth order moment errors are for this rotation offset already far below 0.32 %. Thus a minimum rotation offset of 3.58 percent of the total packet stream length is desirable.

Based on this number, we can predict that a discrete scaling of this AbileneIII packet stream can be computed by summing rotated versions of the original bin count vector up to a scaling factor of about 28. Of course, for high scaling factors, the overhead introduced by using a traffic model diminishes. For high factors it will be better – computationally and for precision – to use a traffic model.

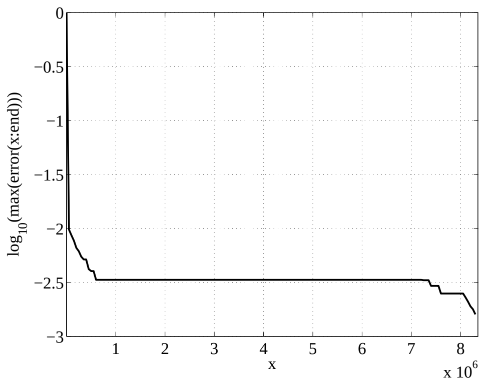


Figure 9. Maximum absolute value of relative second order moment error for rotation offsets larger than x.

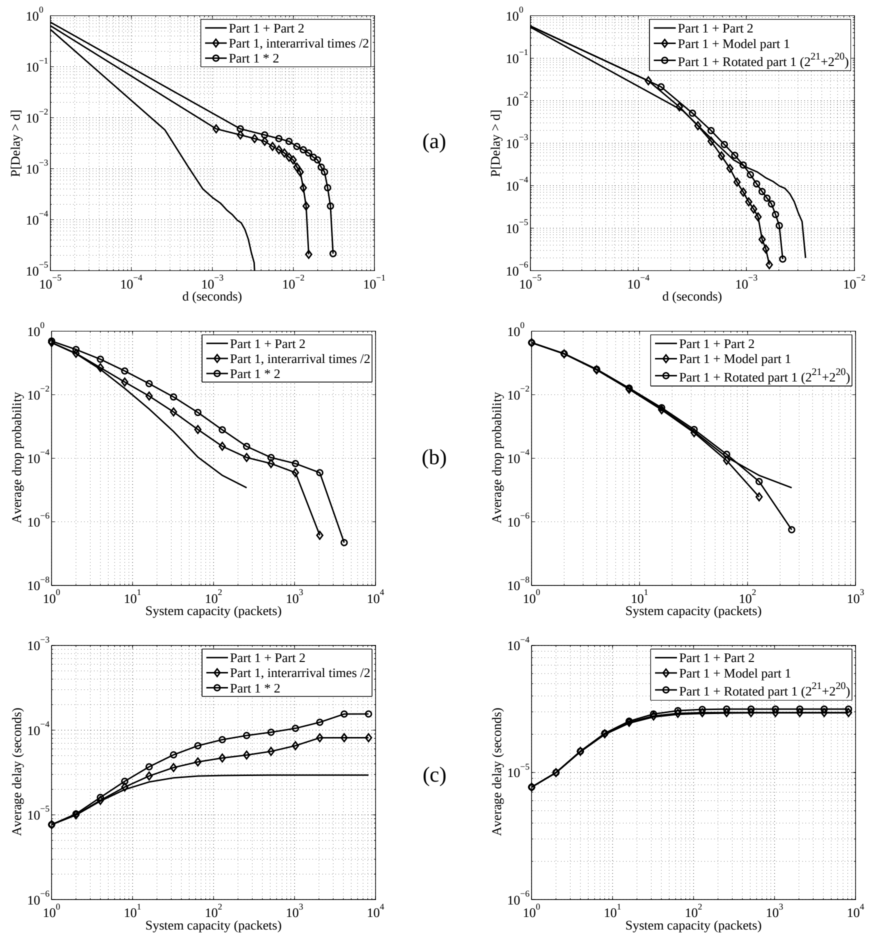


Figure 10. Queuing behavior for different scenarios (randomized intra-bin timestamps).
(a) Delay distribution for infinite queue size, (b) Average drop probability for finite queue size, (c) Average delay for finite queue size.

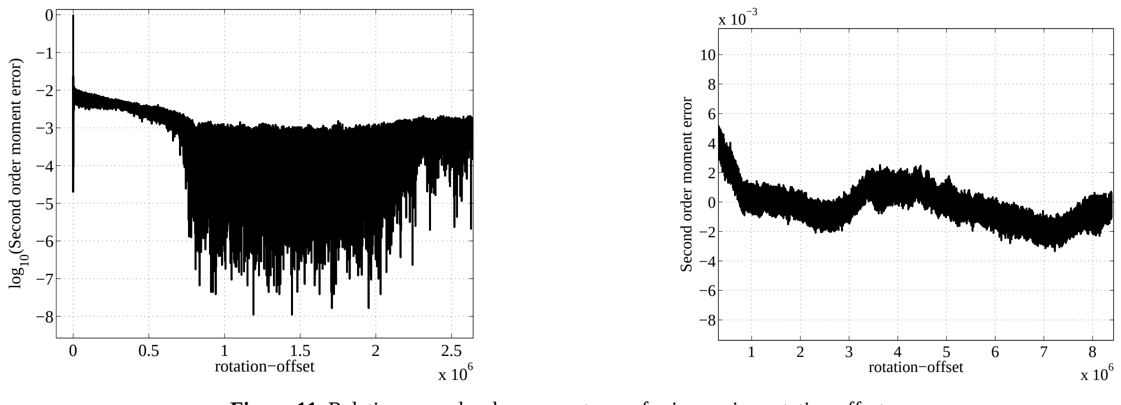


Figure 11. Relative second order moment error for increasing rotation offset.

We remark that obviously a rotation offset of half the vector length is even better in case of the second order moment, resulting in a relative error of approximately 0.16 %. If searching for the optimal rotation offset is considered too time-consuming, choosing the rotation offset at half the vector length is thus a reasonably good choice for doubling the traffic load.

When one wants to scale traffic with non-discrete factors (e.g. 1.3), the traffic model technique will be the only possibility. The moments can then simply be multiplied by the desired factor, followed by generation of the packet stream.

8. CONCLUSIONS

We conclude that techniques like simply doubling bin count or halving interarrival times are, despite their simplicity, not quite accurate in doubling the load inflicted by a packet stream. Especially the correlation between original and added packet streams generates large deviations from the ideal case.
In the study of other techniques, rotating the bin vector to decrease correlation seems to generate good results for low multiplication factors. For larger factors, the correlation increases again, while computational effort also rises. For these large factors, the best results with the smallest effort can be generated by using traffic models. In this paper we used a dyadic lognormal model [10] that fits the mean and second and third order moments. Moments behavior of this generated packet stream equalled the ideal behavior quite good for second and third order moments. As a result, the queuing behavior resembled the rotation technique queuing behavior quite well. An improvement can be expected by also fitting fourth order moments. When non-integer scaling factors are wanted, traffic models have to be used.

Another important remark is the fact that we made abstraction of the packet sizes. Because the packet delays in routers mainly consist of routing delays (lookups in routing tables), packet size plays a minor role here. But if we want to simulate the performance of a switched network, the delay is mostly determined by the transmission delay, which is proportional to the packet size. In this scenario, next to a model for packet interarrival times, an adequate modeling of packet sizes is also necessary.

REFERENCES

[1] G. Iannaccone, C. Diot, I. Graham and N. McKeown. “Monitoring very high speed links.” In *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 267-271, 2001.

[2] A. McGregor, H.W. Braun and J. Brown. “The NLANR Network Analysis Infrastructure.” *IEEE Communications Magazine*, 38(5): 122-128, 2000.

[3] Mark Crovella and Azer Bestavros. “Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes.” *IEEE/ACM Transactions on Networking*, 5(6): 835-846, 1997.

[4] Will E. Leland, Murad S. Taqq, Walter Willinger and Daniel V. Wilson. “On the self-similar nature of Ethernet traffic.” In *Proc. of ACM SIGCOMM*, pages 183-193, San Francisco, California, 1993.

[5] Trang Dang Dinh, Sándor Molnár and Atilla Vidács. “Investigation of Fractal Properties in Data Traffic.” *Journal on Communications*, XLIX:12-18, November-December 1998.

[6] D. Figueiredo, B. Liu, V. Misra and D. Towsley. “On the Autocorrelation Structure of TCP Traffic.” *Computer Networks Journal Special Issue*, 2002.

[7] Vern Paxson and Sally Floyd. “Wide area traffic: the failure of Poisson modeling.” *IEEE/ACM Transactions on Networking*, 3(3): 226-244, 1995.

[8] W. Willinger and V. Paxson. “Where Mathematics meets the Internet.” *Notices of the American Mathematical Society*, 45(8): 961-970, 1998.

[9] K. Park, G. Kim and M. Crovella. “On the effect of traffic self-similarity on network performance.” In *Proc. of the SPIE Int. Conference on Performance and Control of Network Systems*, Nov. 1997.

[10] J. Potemans, J. Theunis, B. Van den Broeck, Y. Guan, E. Van Lil and A. Van de Capelle. “Modelling Fractal Internet Traffic: Additive Dyadic Superposition of Lognormal Distributions.” In *Proc. of The IASTED International Conference on Communication and Computer Networks (CCN2004)*, MIT, Cambridge, MA, USA, November 2004.

[11] K. Vishwanath and A. Vahdat. “Swing: Generating Representative High-Speed Packet Traces.” In *Proc. of ACM SIGCOMM*, Philadelphia, PA, USA, 2005

[12] P. Kamath, Kun-chan Lan, J. Heidemann, J. Bannister and J. Touch. “Generation of High Bandwidth Network Traffic Traces.” In *Proc. of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Fort Worth, Texas, USA, October 2002

[13] J. Potemans, J. Theunis, P. Leys, B. Van den Broeck, E. Van Lil and A. Van de Capelle. “Advanced Traffic Modeling: Fitting Third Order Moments.” In *Proc. to IEEE Global Conference on Communications GLOBECOM2003*, San Francisco, CA, USA, December 2003.