A web-based tool for advanced statistical analysis of network traffic measurements*

Costas Courcoubetis† and Vasilios A. Siris
Institute of Computer Science (ICS)
Foundation for Research and Technology - Hellas (FORTH)
P.O. Box 1385, GR 711 10 Heraklion, Crete, Greece
E-mail: {courcou, vsiris}@ics.forth.gr

June 26, 2000

Abstract

We present a web-based tool for advanced statistical analysis of network traffic measurements. The tool consists of an interface and software modules implementing advanced statistical analysis procedures. The interface provides a flexible environment through which a user, using a Java enabled web browser, can create, modify, save, and execute experiments. The experiments are executed by software modules implementing analysis procedures based on a theory of statistical multiplexing a large number of bursty traffic streams while guaranteeing some Quality of Service (QoS); the input to the software modules are traces that contain measurements of actual network traffic. Questions that can be investigated using the tool include the maximum link utilization when some QoS is guaranteed, how this utilization is affected by the link buffer and traffic shaping, the acceptance region when some QoS is guaranteed, and the token bucket parameters for a traffic stream.

Keywords: Traffic analysis, network management, network dimensioning, token/leaky bucket, web-based tool

1 Introduction

Recent advances in protocol development and technology are creating the necessary building blocks for transforming the Internet from a pure best-effort services network into a network capable of supporting Quality of Service (QoS) guarantees. QoS guarantees, along with bounds to the traffic the user is allowed to send into the network form the Service Level Agreement (SLA) between the user and the network provider. The aforementioned capabilities increase the flexibility of networks in handling traffic with different performance requirements, but they also increase the complexity of network management and dimensioning.

Important questions related to the management and dimensioning of networks that support QoS guarantees include the following: What combination of users can a network provider accept while ensuring some target level of performance? How many more users can the provider accept if he increases the capacity of his network or shapes incoming traffic? How can a user select the parameters of the traffic contract or service level agreement (SLA) with his provider?

An objective in managing such networks is the efficient use of network resources shared through statistical multiplexing, while guaranteeing some QoS. Traditional approaches to network management such as measuring the average load in intervals of the order of minutes are not adequate, since they fail to capture traffic burstiness, which is important when some QoS is guaranteed. Approaches based on queueing theory are also inadequate, since they require elaborate traffic models and cannot be effectively applied in the context of large multi-service

*The latest version of the tool is available at `http://trace.ucnet.uoc.gr`
†Also with the Athens University of Economics and Business.

networks. Furthermore, evidence of self-similar or fractal behavior of network traffic (e.g., see [7, 8]) has rendered traditional traffic source models inadequate.

In addition to enabling networks to offer QoS guarantees, advances of technology have enabled the collection of detailed traffic statistics at very high speeds: 155 Mbps, 622 Mbps, and higher; see [1]. At such high speeds the amount of measurement data is huge. Hence, inevitable questions posed by both researchers and engineers is whether all this data is required, and if not, which measurements are most important and how can they be used for network management[1].

In this paper, we present the functionality and design of a web-based tool for advanced statistical analysis of real network traffic measurements. The tool consists of the following two components:

- a web-based user interface
- software modules implementing advanced statistical analysis procedures

The interface provides a flexible and interactive environment through which a user, using a Java enabled web browser, can create, modify, save, and execute experiments. The results of the experiments can be viewed graphically, with the capability of presenting in the same graph the results from more than one experiments of the same type.

The analysis modules implement procedures that are based on a recent theory of statistical multiplexing of a large number of bursty traffic streams, while guaranteeing some QoS. The approach does not involve traffic models, but relies on actual traffic measurements from which the effective bandwidth of a stream, which is a scalar that measures the relative amount of resources used by the stream, can be estimated. In addition to answering important questions concerning network management and dimensioning, the approach can give guidelines for what traffic statistics, and in particular the time granularity of traffic measurements, are important for evaluating the performance of a link.

The rest of this paper is structured as follows. In Section 2 we discuss typical questions related to the management and dimensioning of networks that support QoS. In Section 3 we describe the functionality and the design of the web-based tool for traffic analysis, and in Section 4 we describe the software modules that implement the advanced traffic analysis procedures. In Section 5 we conclude the paper and identify related research activities we are currently pursuing. In Appendix A we give some theoretical backround and a brief overview of the procedures implemented by one of the software modules, namely the `msa` module.

2 Questions concerning network management

Next we discuss some typical questions concerning the management and dimensioning of networks that carry bursty traffic and guarantee some level of performance. Our discussion considers a single link which is statistically shared by a number of bursty traffic streams of various types (e.g., web, video, and voice traffic). The performance or Quality of Service (QoS) measure that we consider is the probability of traffic being delayed more than some maximum value[2]. Finally, the link has some amount of resources (capacity and buffer), and services packets according to some scheduling discipline (e.g., First-Come-First-Served - FCFS, priorities).

Question 1: Resource usage

What is the amount of resources used by each traffic stream? For an aggregate traffic flow, what is the contribution of each individual flow (e.g., from a subnet or protocol) to the total amount of resources required for the aggregate flow?

[1]See presentations at the Internet Statistics and Metrics Analysis (ISMA) workshop on passive measurement data and analysis at `http://www.caida.org/outreach/isma/9901/`

[2]The proportion of time a buffer of size B is full (probability of overflow) is equivalent to the probability of traffic being delayed more than $D = B/C$ in an infinite buffer served at rate C.

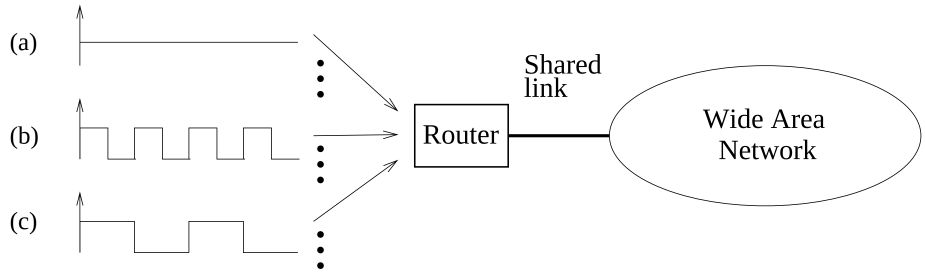

Figure 1: A stream's resource usage depends not only on its statistical characteristics, but also on the link resources, the QoS guarantees, and the characteristics of the other traffic it is multiplexed with.

Traditionally, the above questions are answered by measuring the average load produced by each flow in intervals of the order of minutes. Such an approach is appropriate for best-effort networks, but is inaccurate for networks with QoS guarantees. For such networks, resource usage depends not only on the statistics of the stream, but also on the multiplexing that occurs at the link.

As an example, consider a link that multiplexes three types of traffic streams while ensuring some QoS, Figure 1. All three stream types have the same average rate. However, type (a) is smooth (constant rate) whereas the other two are bursty with different duration of "on" and "off" periods. What amount of resources does each stream type require? Both theory and experimentation indicate that the answer to this question depends on the link resources, the QoS guarantees, and the characteristics of the multiplexed traffic. Hence, when the capacity of the link is small, it might be the case that resource usage for stream type (a) is smaller than that for type (b), which in turn is smaller than that for type (c). For a larger capacity, it might be the case that resource usage for stream types (a) and (b) are the same, and smaller than that for type (c). Finally, for even larger capacities or when the QoS guarantees are loose, it might be the case that all three stream types have the same resource usage.

Question 2: Traffic measurements

What is the size of the time interval for traffic measurements (number of bytes or cells) so that all the statistics that affect a link's performance are captured?

At one extreme, one can capture traces of individual packets. In addition to producing a very large amount of data, particularly for high speed links, such fine statistics can contain more information than what is necessary for evaluating the performance of a link. An alternative is to measure the load in consecutive intervals of the same length. A longer measurement interval results in less amount of measurement data, but also in less statistical information, since fluctuations are averaged within each interval. The above question refers to the maximum length of the time interval so that all the statistics that affect a link's performance are captured.

Question 3: Acceptance region and link utilization

Given the link resources (capacity and buffer), what combinations of traffic types can be multiplexed while ensuring a target QoS? These combinations of traffic types form the *acceptance region.*

A related question is the following: Given the link resources and the traffic mix (traffic types and percentage of each type), what is the maximum link utilization that can be achieved (equivalently, the maximum number of streams that can be multiplexed), while ensuring a target QoS? How much can the link's utilization increase when its resources increase by some amount?

Question 4: Quality of Service

Given the link resources, the traffic mix, and the link utilization (number of multiplexed streams), what is the level of QoS offered? How much does the QoS improve when the link's resources increase by some amount?

Question 5: Resource dimensioning

Given the number and combination of traffic streams, what is the minimum amount of resources (capacity and buffer) required to guarantee a target QoS?

Question 6: Traffic shaping

What effect does traffic shaping have on the link's multiplexing capability? How does the effect of traffic shaping depend on the link's buffer size?

Question 7: Scheduling discipline

What effect does the scheduling discipline have on the link's multiplexing capability?

Question 8: Leaky bucket parameter selection

What leaky or token bucket parameters should be selected for a particular flow? How does traffic shaping affect the value of these parameters? For the wide area link connecting a large organization to the Internet, how do the token bucket parameters change throughout the day.

3 Web-based tool for traffic analysis: functionality and architecture

The interface (Figure 2) provides a flexible environment through which a user, using a Java enabled web browser, can create, modify, save, and execute *experiments*. Experiments can be of the following type:

1. Buffer Overflow Probability (BOP) of a link for a specific traffic mix.
2. Maximum link utilization for a specific traffic mix, while satisfying a target BOP.
3. Combinations of two types of traffic streams that satisfy a target BOP (this forms the acceptance region).
4. Token bucket's leak rate vs. bucket size trade-off for a traffic stream (called indifference curve). The indifference curve can be calculated both for the case where all traffic is conforming and for the case where some percentage of the traffic can be non-conforming.

The link model considered is that of a single buffer of size B serviced First-Come-First-Served (FCFS) at rate C, Figure 3. The buffer is fed by a number of bursty traffic streams, which are given by actual traffic traces. Before entering the buffer, the traffic streams can go through one or more filters ("Fs" in Figure 3). Possible filters include traffic shaping, peak rate policing, and leaky bucket policing.

The tool currently contains traces of actual MPEG compressed video traffic (see [9]), and measurements taken on an operational IP network.[3]

Finally, the tool contains a number of predefined experiments for each experiment type. Hence, the user can start by executing these predefined experiments, and gradually modify the parameters to create new experiments. An example definition of an experiment is shown in Table 1.

[3] In particular, from the University of Crete's wide area link with the Greek Research and Technology network (GRnet). See [3] and `http://trace.ucnet.uoc.gr` for more details.

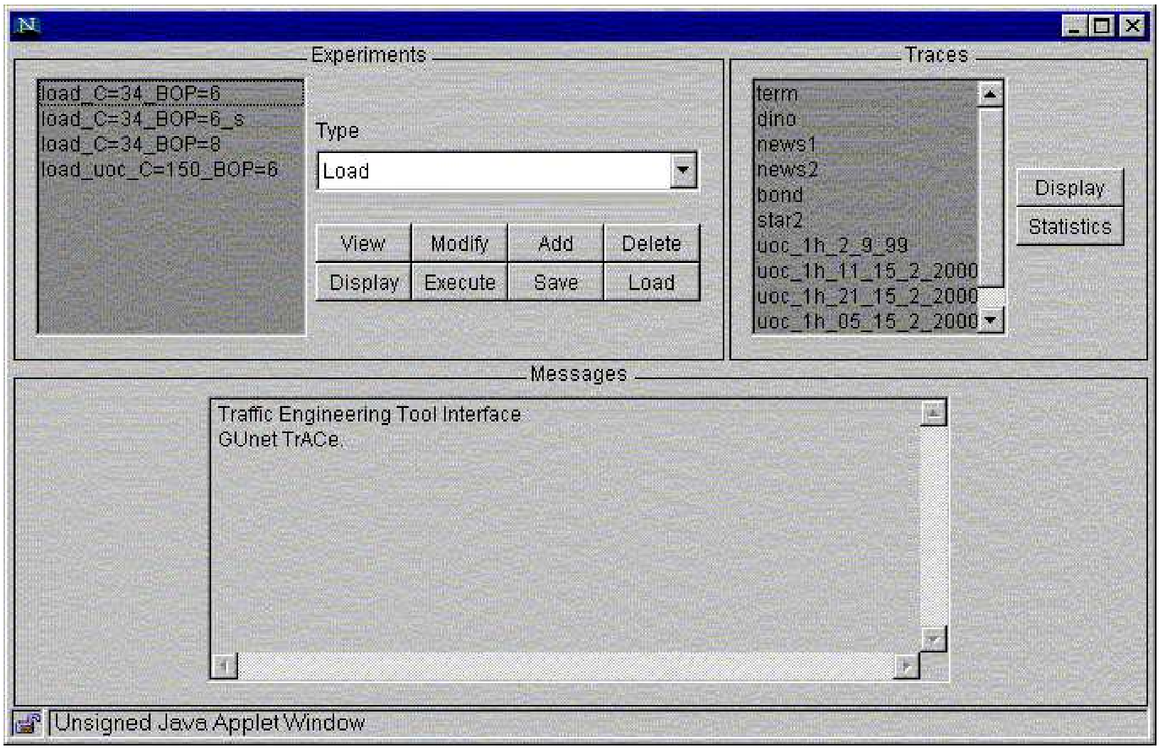

Figure 2: The tool's user interface. The upper left box is the experiments box, from which the user can create, save, and execute experiments. The upper right box shows the available traffic traces, including miscellaneous statistics for the traces. The lower box displays status messages.

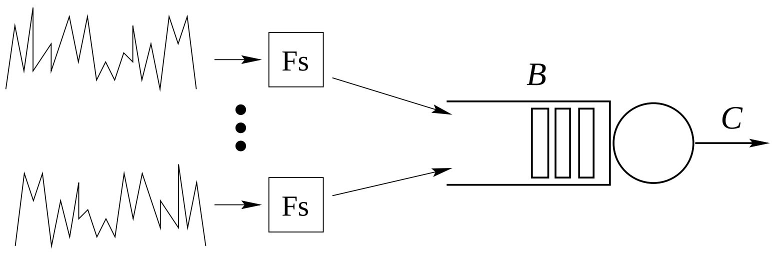

Figure 3: Link model. A buffer of size B serviced First-Come-First-Served (FCFS) at rate C is fed by a number of bursty traffic streams, which are given by actual traffic traces. Before entering the buffer, the traffic streams can go through one or more filters ("Fs" in the figure).

Experiment type	load (maximum utilization)
Traffic stream	star2
Filters	none
C	34 Mbps
B	from 5 msec to 40 msec, step 5 msec
$-\log_{10}(BOP)$	6
Method	Many source + Bahadur Rao

Table 1: Example definition of an experiment. The above experiment produced the lower curve in Figure 5. Observe that the buffer size is measured by the corresponding queueing delay, computed from the ratio B(in Mbits)/C(in Mbits/msec).

3.1 Graphical display of results

In this section we present examples of experiments that a user can run. The experiments we present are predefined, and available to users when they first download the tool's interface. By modifying the parameters of these predefined experiments, the user can create new experiments.

The results of the experiments are displayed graphically. For example, with the third experiment type mentioned previously, the acceptance region for two traffic stream types will be displayed, Figure 4.

Furthermore, the tool can display the results from two experiments of the same type in the same graph. This capability can be used to view, e.g. the effects of traffic shaping on the maximum link utilization, Figure 5, and the leaky bucket parameters for different periods of the day, Figure 6 .

3.2 Architecture

The architecture of the traffic analysis tool is shown in Figure 7. The client interface (written in Java) runs on the user's browser, and sends requests to a server (also written in Java) that runs on a high performance workstation running FreeBSD. The requests can be to save, load, or execute experiments. In the latter case, the server launches the execution of shell scripts which in turn run the appropriate software module, namely the `msa` or `lb` modules (both written in C) which are described in the next section. When the execution of an experiment is completed, the server sends the results back to the client interface, which displays the results in a graph located in a new window (i.e. different from the user interface window).

4 Traffic analysis procedures

The analysis of traffic measurements is performed primarily by two modules: the first, called `msa`, implements the analysis based on the effective bandwidths and many sources asymptotic, and the second, called `lb`, computes the token (or leaky) bucket parameters. Both modules take input from a trace file which contains the load (number of bytes or cells) in consecutive intervals (epochs) of fixed duration. A sample trace file is shown below:

```
# epoch_in_msecs = 10
# bits_per_info_unit = 8
737
584
551
657
...
```

The first two lines define the epoch duration and the size (in bits) of the units of load, respectively. Hence, according to the above trace the stream produced 737 bytes (since one byte contains 8 bits) in the first epoch ($0 - 10$ msec), 584 bytes in the second epoch ($10 - 20$ msec), etc.

Both modules, along with a detailed manual and other related information, are publically available as stand-alone programs at [10]. In [3] we further discuss the two modules and present case studies where we apply the modules to traces of actual network traffic.

In addition to the two aforementioned modules, we have written a number of supportive programs (e.g. for traffic smoothing and policing) and scripts for executing the same function for a range of link parameters. Using the scripts we can, e.g. run the function for computing the maximum utilization for a range of buffer sizes, thus producing the data for plotting the maximum utilization as a function of buffer size.

In the next two subsections we describe the procedures and functionality of the `msa` and `lb` modules.

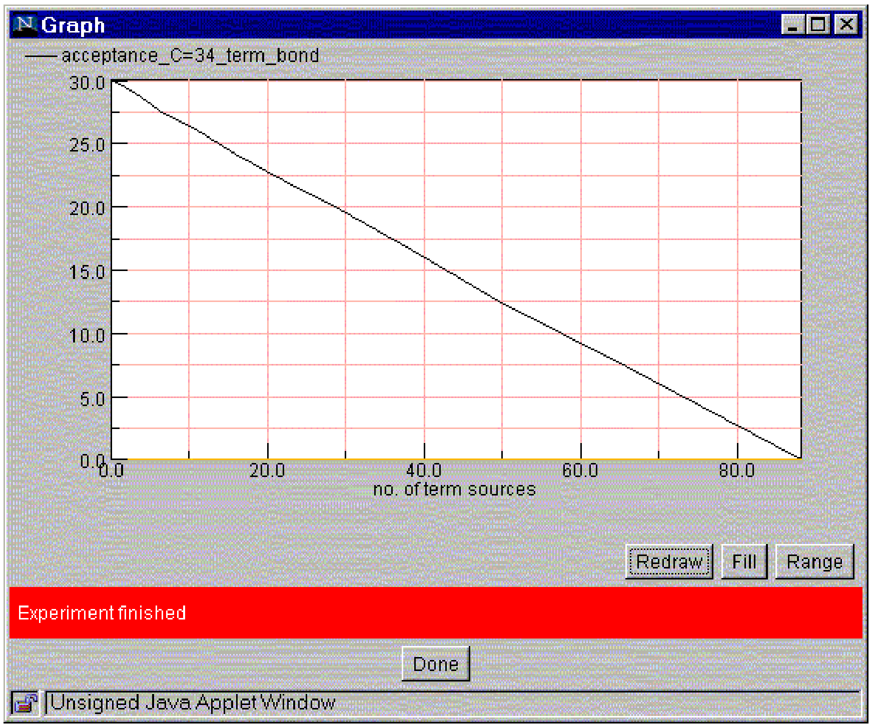

Figure 4: Acceptance region for two types of traffic streams, namely "Terminator" and "James Bond" streams (both are MPEG compressed video traffic); the two axes depict the number of streams of each type. $C = 34$ Mbps, $B = 4$ msec, and $BOP = 10^{-5}$.

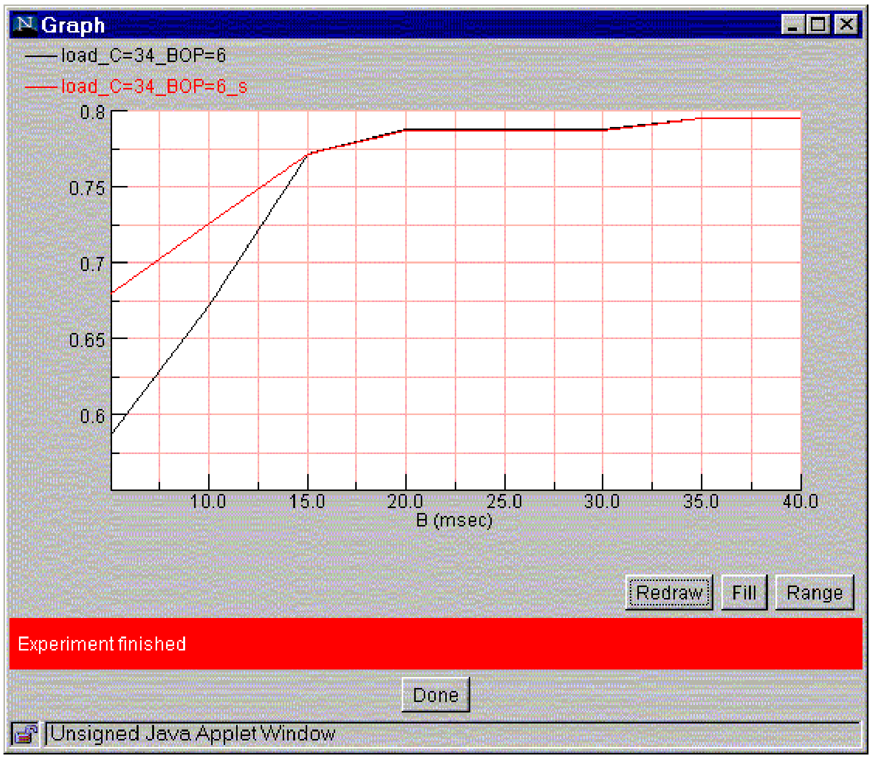

Figure 5: Maximum link utilization experiment: effect of traffic shaping on the maximum utilization. $C = 34$ Mbps, $BOP = 10^{-6}$, "Star Wars" traffic stream. The top (for buffer sizes less than 15 msec) curve corresponds to a smoothed traffic stream obtained by averaging traffic of the original stream in consecutive time intervals of duration 80 msec.

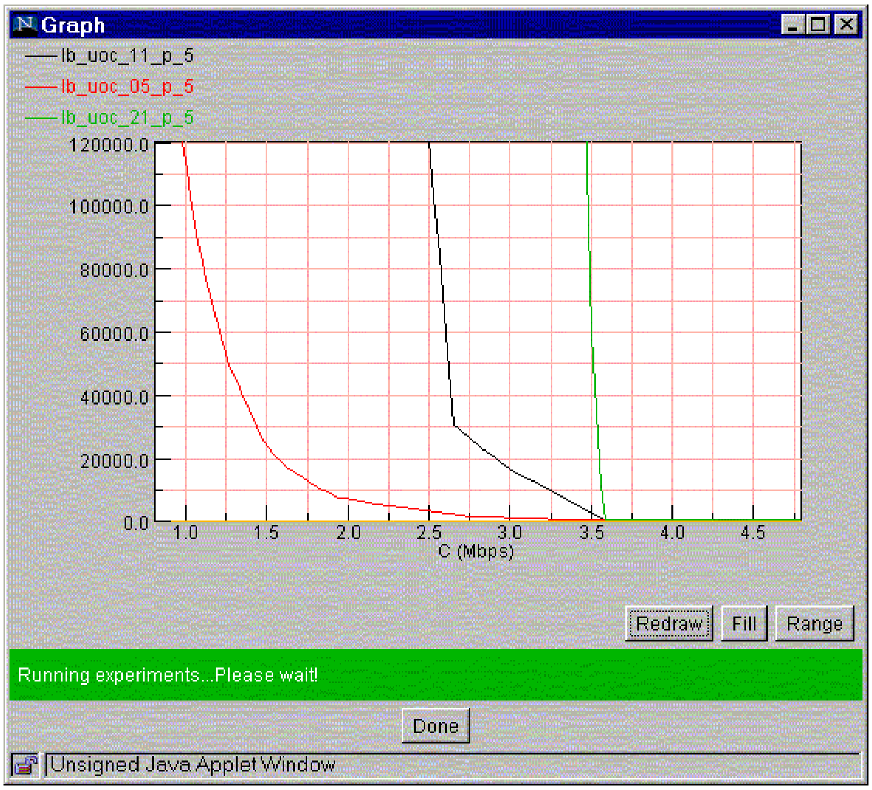

Figure 6: Leaky bucket experiment: leaky bucket parameters for different periods of the day, when up to 10^{-5} of the traffic is non-conforming. From left to right, the curves are for traces of the University of Crete's incoming traffic collected on 15/2/2000 over a one hour period at times 05:00, 11:00, and 21:00, respectively.

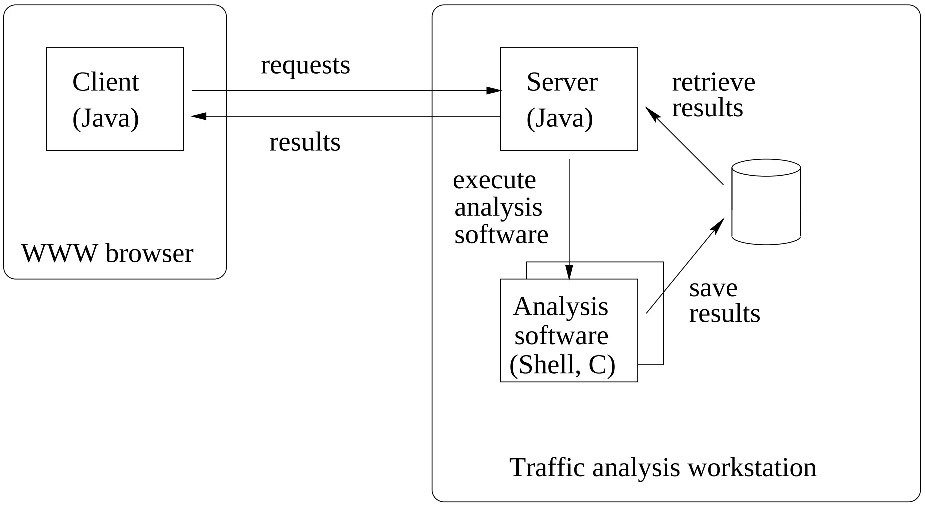

Figure 7: Architecture of the web-based traffic analysis tool. The client sends requests to the server. If the request refers to the executing an experiment, the corresponding software module is launched. The results of the experiment are returned to the client. We note that our implementation allows the server and analysis software to be executed on different workstations.

4.1 The msa module

The `msa` module is based on a recent theory of multiplexing a large number of bursty traffic streams, while guaranteeing some level of performance [6, 5, 2]. This work considers the effective bandwidth of a stream as a measure of resource usage of that stream. The effective bandwidth of a stream depends on the context of the stream (see the related discussion of Question 1 in Section 2) through only two parameters s, t. These parameters can be taken to characterize the operating point of a link. Furthermore, periods of the day during which the traffic mix remains relatively constant can be characterized by a single pair, which can be computed off-line from traffic measurements [4]. In Appendix A we present some theoretical backround and describe the procedures implemented by the `msa` module.

The `msa` module has the following five functions:

1. Calculates the effective bandwidth (measure of resource usage) for a given link operating point (expressed through parameters s, t) and traffic mix (source types and percentage of each type).

2. Calculates the buffer overflow probability (BOP) for a given link capacity, buffer, number of sources, and traffic mix.

3. Calculates the maximum load (equivalently, the maximum number of sources) for a link of given capacity, buffer, and traffic mix, while ensuring a maximum buffer overflow probability.

4. Calculates the minimum buffer size for a link with a given capacity, number of sources and traffic mix, such that a maximum buffer overflow probability is satisfied.

5. Calculates the minimum capacity for a link with a given buffer size, number of source and traffic mix, such that a maximum buffer overflow probability is satisfied.

By running the above functions for a range of link parameters we can compute (i) the buffer overflow probability (BOP) as a function of buffer size (for fixed capacity, traffic mix and load), (ii) the maximum load as a function of buffer size (for fixed capacity, overflow probability, and traffic mix), see Figure 5, and (iii) the acceptance region (for fixed capacity, buffer, and overflow probability), see Figure 4.

4.2 The lb module

The `lb` module computes the token (or leaky) bucket parameters for a particular trace file when all traffic is to be conforming or when some percentage of the traffic is allowed to be non-conforming. A token bucket (see Figure 8) consists of a token pool of size b (bucket size) which fills at rate r (token rate) measured in tokens per second. The token bucket is used to police a traffic stream in the following way: While there are enough tokens, a number of tokens equal to the size of the packet are removed from the token bucket. In this case the packet is said to be conforming. On the other hand, if the number of token's in the bucket is less than the size of the packet, then the packet is non-conforming.

For a particular trace, there is not a single pair of token bucket parameters, but a set of such pairs which form the indifference curve, Figure 9. The indifference curve of a stream is convex, intersects the rate axis (horizontal axis in Figure 9) at the peak rate p of the stream, and increases abruptly as the token rate approaches the mean rate m of the stream.

The indifference curve of a stream can be numerically computed in the following way: For values of r ranging from the mean rate to the peak rate, we emulate a single queue with infinite buffer being fed with traffic from the trace file, and find the maximum backlog b such that all or a minimum percentage of the traffic is conforming. With the above procedure we obtain the pairs (r, b) that belong to the trace's indifference curve.

If all traffic is to be conforming, then the maximum backlog b can be found in a single pass of the trace file. However, when some percentage of the traffic is allowed to be non-conforming,

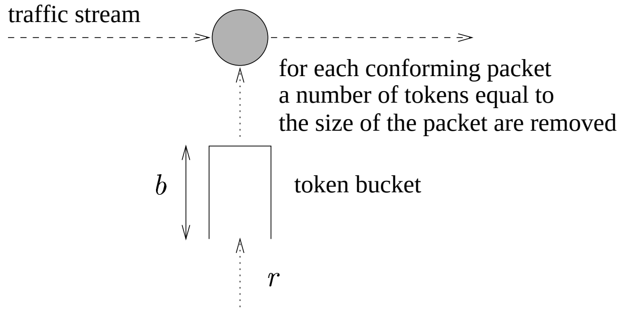

Figure 8: Token bucket policing. A packet is conforming if when it arrives the number of tokens in the token bucket is more than the packet size, else it is non-conforming.

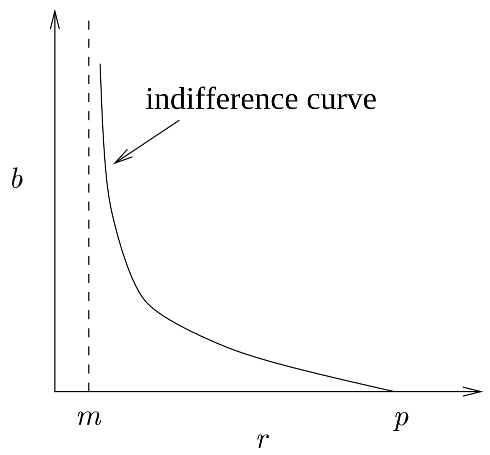

Figure 9: The indifference curve is convex, intersects the rate (horizontal) axis at the stream's peak rate p, and increases abruptly as the token rate approaches the stream's mean rate m.

then for each value of r we must search for the minimum value of b such that the percentage of backlog traffic above b is equal to the target percentage of non-conforming traffic. Calculating this percentage requires one pass of the trace file for each value of b that is searched. Because the percentage of non-conforming traffic is monotone with b, the minimum b can be found using a binary search.

5 Conclusion and related work

We have presented a web-based tool for advanced analysis of network traffic measurements.[4] The tool consists of a flexible user interface (written in Java), and software modules (written in C), that implement advanced statistical analysis procedures based on a theory of multiplexing a large number of bursty traffic streams while guaranteeing some QoS. In addition to assisting managers in operating and dimensioning their networks more efficiently, the tool can serve as a teaching tool for hand-on experience with traffic analysis methods.

The development of the web-based tool for advanced statistical analysis is part of our activities in traffic measurement and analysis, which are being performed in collaboration with the Communication and Networks Center at the University of Crete.[5] The objectives of this work, in addition to the development of the tool presented in this paper, includes the deployment of a flexible platform for passive measurement of network traffic from an operational network and the development of web-based tools for requesting and displaying on-line traffic statistics, such as the composition (e.g. application, protocol), origin (e.g. national or international link), and destination (e.g. department, subnet) of the traffic. Finally, the platform is being enhanced to support active traffic measurements and to serve as a testbed for implementing usage-based charging experiments with actual network traffic.

Acknowledgements

We thank Antonis Dimakis who wrote the initial version of the tool's user interface.

References

[1] CoralReef. `http://www.caida.org/tools/measurement/coralreef/`.

[2] C. Courcoubetis, F. P. Kelly, and R. Weber. Measurement-based usage charges in communications networks. Technical Report 1997-19, Statistical Laboratory, University of Cambridge, 1997. To appear in *Operations Research.*

[3] C. Courcoubetis and V. A. Siris. Measurement and analysis of real network traffic. Technical Report No. 252, ICS-FORTH, March 1999.

[4] C. Courcoubetis, V. A. Siris, and G. D. Stamoulis. Application of the many sources asymptotic and effective bandwidths to traffic engineering. *Telecommunication Systems*, 12:167–191, December 1999. A shorter version appeared in *Proc. of ACM SIGMETRICS'98/PERFORMANCE'98.*

[5] C. Courcoubetis and R. Weber. Buffer overflow asymptotics for a switch handling many traffic sources. *J. Appl. Prob.*, 33:886–903, 1996.

[6] F. P. Kelly. Notes on effective bandwidths. In F. P. Kelly, S. Zachary, and I. Zeidins, editors, *Stochastic Networks: Theory and Applications*, pages 141–168. Oxford University Press, 1996.

[4] The latest version of the tool is available at `http://trace.ucnet.uoc.gr`

[5] More information on these activities and other web-based tools are available at `http://trace.ucnet.uoc.gr`.

[7] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic. In *Proc. of ACM SIGCOMM'93*, pages 183–193, Ithaca, NY, USA, September 1993.

[8] V. Paxson and S. Floyd. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Trans. on Networking*, 3(3):226–244, June 1995.

[9] O. Rose. Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems. Technical Report 101, University of Wuerzburg, February 1995.

[10] V. A. Siris. Large deviation techniques for traffic engineering. `http://www.ics.forth.gr/netgroup/msa/`.

A Effective bandwidths and many sources asymptotic: theory and application

Next we present the theory and application of the effective bandwidths and the many sources asymptotic [6, 5, 2, 4]. The effective bandwidth of a stream of type j that produces load $X_j[0,t]$ in a time interval t is defined as

$$\alpha_j(s,t) = \frac{1}{st} \log E\left[e^{sX_j[0,t]}\right], \tag{1}$$

where the parameters s and t characterize a link's operating point and depend on the context of the source, i.e., the link resources and the characteristics of the multiplexed traffic. Specifically, the space parameter s (measured in, e.g., Mbit^{-1}) indicates the degree of multiplexing and depends, among others, on the size of the peak rate of the multiplexed streams relative to the link capacity: For links with capacity much larger than the peak rate of the multiplexed streams, s tends to zero and the effective bandwidth approaches the mean rate, while for links with capacity not much larger than the peak of the streams, s is large and the effective bandwidth approaches the peak rate $X[0,t]/t$, measured over an interval of duration t. On the other hand, the time parameter t (measured in, e.g., seconds) corresponds to the most probable duration of the buffer busy period prior to overflow.

Computation of the buffer overflow probability (BOP)

If N streams are multiplexed in a link with capacity C and buffer B, and ρ_j is the percentage of streams of type j, then the parameters s,t can be computed from the following formula

$$NI = \inf_t \sup_s F(s,t), \tag{2}$$

where

$$F(s,t) = s(B+Ct) - stN \sum_j \rho_j \alpha_j(s,t).$$

The buffer overflow probability (BOP) can then be approximated by

$$BOP \approx e^{-NI}. \tag{3}$$

The many sources asymptotic can be improved using the Bahadur-Rao theorem. With the Bahadur-Rao improvement the buffer overflow probability is approximated by

$$BOP \approx e^{-NI - \frac{1}{2}\log(4\pi NI)},$$

where NI is computed as before using (2).

Let T be the total duration of the trace. The expectation in (1) can be approximated by the empirical average, hence the effective bandwidth can be approximated by (for simplicity we assume that T is integer multiples of t)

$$\tilde{\alpha}_j(s,t) = \frac{1}{st} \log \left[\frac{1}{T/t} \sum_{i=1}^{T/t} e^{sX_j[(i-1)t,it]} \right] .$$

The solution of (2) involves two optimization procedures: the first consists of finding, for a fixed value t, the maximum $F^*(t) = \max_s F(s,t)$, and the second consists of finding the minimum $NI = \min_t F^*(t)$.

The maximization $F^*(t) = \max_s F(s,t)$ can be numerically solved in an efficient manner by taking into account that the logarithmic moment generating function $st\alpha_j(s,t) = \log E[e^{sX_j[0,t]}]$ is convex in s, whereas $s(B+Ct)$ is linear in s . Due to this, $F_t(s) = F(s,t)$ is a unimodal function of s and the maximizer is unique. Hence, to find $F^*(t) = \max_s F(s,t) = \max_s F_t(s)$ one can start from an initial "uncertainty" interval $[s_a, s_b]$ that contains the maximum (for this to be the case it is sufficient that for some $x \in [s_a, s_b]$ we have $F_t(x) > F_t(s_a)$ and $F_t(x) > F_t(s_b)$), and decrease the uncertainty interval using a *golden section* search as follows:

1. Given the interval $[s_a, s_b]$, two trial points s_l, s_r are selected such that $s_r - s_a = s_b - s_l = g(s_b - s_a)$, where $g = 0.618$ is (approximately) the *golden ratio*.
2. We evaluate $F_t(s_l)$ and $F_t(s_r)$, and identify three cases:
   (a) if $F_t(s_l) > F_t(s_r)$ the interval becomes $[s_a, s_r]$,
   (b) if $F_t(s_l) < F_t(s_r)$ the interval becomes $[s_l, s_b]$, and
   (c) if $F_t(s_l) = F_t(s_r)$ the interval becomes $[s_l, s_r]$.
3. Steps 1 and 2 are repeated until the uncertainty interval has length less than some small value ϵ.

The golden section search is the limit (for a large number of steps) of the *Fibonacci search*, which minimizes the maximum number of steps needed to reduce the uncertainty interval to some prescribed length.

Unlike the maximization procedure for $\max_s F(s,t)$, there is no general property for $F^*(t)$ that we can take advantage of in order to perform the minimization $\min_t F^*(t)$ in an efficient manner.[6] For this reason, the minimization is solved by linearly searching the values of t in the interval $[0, \kappa\tau]$ with granularity equal to one epoch τ. The value of κ is determined empirically and depends on the buffer size: the extremising value of t is larger for larger buffer sizes.

The above describe the numerical solution of the `infsup` formula (2) which, using (3), implements the second function of the `msa` module described in Section 4.1.

To select the time granularity of traffic measurements (i.e., the length of the interval for measuring the number of bytes or cells) so that all statistics that are important for buffer overflow are captured, one proceeds as follows. The minimizing value of t in (2) is computed as described above. Obtaining the value $t = \tau$ (the duration of one epoch) indicates that buffer overflow occurs on time scales less than (or equal) to τ. In this case the load measurements are too coarse, and the measurement interval must be decreased so that the minimizing value of t is a few times τ.

Computation of the maximum number of multiplexed streams

Now we describe the implementation of the third function of the `msa` module described in Section 4.1. The maximum utilization, or equivalently the maximum number of streams, for

[6]Furthermore, experiments have shown that $F^*(t)$ can have more than one local maxima.

a given amount of link resources (capacity C and buffer B), traffic mix $\vec{\rho} = \{\rho_1, \rho_2, \ldots\}$, and target overflow probability $e^{-\gamma}$ can be computed by solving the following equation

$$N = \inf_t \sup_s G(s,t)\,, \tag{4}$$

where

$$G(s,t) = \frac{s(B+Ct)-\gamma}{st\sum_j \rho_j \alpha_j(s,t)}\,.$$

As was the case for $F(s,t)$ in (2), $G(s,t)$ is also a unimodal function of s with a unique maximizer, hence $G^*(t) = \max_s G(s,t)$ can be solved using a golden section search.

To use the Bahadur-Rao improvement, the maximum number of multiplexed sources is computed using (4) after replacing γ with $\gamma_{\text{B-R}} = \gamma - \frac{1/2\log(4\pi\gamma)}{1+1/(2\gamma)}$.

Computation of the minimum buffer size

The fourth function of the `msa` module, i.e., the computation of the minimum buffer size such that a maximum buffer overflow probability is guaranteed can be computed by solving the following equation:

$$B = \sup_t \inf_s K(s,t)\,, \tag{5}$$

where

$$K(s,t) = \frac{stN\sum_j \rho_j \alpha_j + \gamma}{s} - Ct\,.$$

As was the case for $F(s,t)$ in (2), $G(s,t)$ is also a unimodal function of s with a unique minimizer, hence $K^*(t) = \min_s K(s,t)$ can be solved using a golden section search.

To use the Bahadur-Rao improvement, in (5) we replace γ with $\gamma_{\text{B-R}} = \gamma - \frac{1/2\log(4\pi\gamma)}{1+1/(2\gamma)}$.

Computation of the minimum capacity

Finally, the fifth function of the `msa` module, i.e., the computation of the minimum capacity such that a maximum buffer overflow probability is guaranteed can be computed by solving the following equation:

$$C = \sup_t \inf_s H(s,t)\,, \tag{6}$$

where

$$H(s,t) = \frac{stN\sum_j \rho_j \alpha_j + \gamma}{st} - \frac{B}{t}\,.$$

As was the case for $F(s,t)$ in (2), $H(s,t)$ is also a unimodal function of s with a unique minimizer, hence $H^*(t) = \min_s H(s,t)$ can be solved using a golden section search.

To use the Bahadur-Rao improvement, in (6) we replace γ with $\gamma_{\text{B-R}} = \gamma - \frac{1/2\log(4\pi\gamma)}{1+1/(2\gamma)}$.