

A Transaction-Level Tool for Predicting TCP Performance and for Network Engineering

Jean Walrand

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720
wlr@eecs.berkeley.edu

Abstract

Traditional traffic engineering for communication networks assumes that the sources generate traffic with known statistics. Using simulation and/or analysis, one then studies the quality of service that the network offers to these traffic streams.

However, most of the traffic in the Internet is TCP-based. TCP, the Transmission Control Protocol, regulates the delivery of packets to correct errors and avoid saturating the destination of the packets or routers inside the network. Accordingly, the traffic that a TCP source generates is not specified a priori. Instead, this traffic depends in an essential way on the QoS that the network provides.

In this paper, we propose a simple model and we use it to predict the QoS that a network provides to TCP connections. The tool can compute the probability that each connection gets at least some given throughput. Conversely, the tool can be used to design the network to ensure that every connection gets a given throughput with a specified probability. The tool incorporates uncertainties about the load on the network and about the model.

1 Introduction

Network engineering tools for the Internet include measurement systems, simulation packages, and analytical methods and results. The tools should provide guidelines on how to expand network capacity, design new networks, introduce classes of service, provision support for voice over IP (the Internet Protocol), estimate reliability, and assess the potential of new technologies.

Existing tools have serious limitations. Measurement systems are useful to evaluate the performance measures of an existing network. As such, these systems can be used to identify bottlenecks and to detect when the network must

be upgraded. The measurement systems can also help track the evolution of network traffic and performance. However, it is difficult to extract more general conclusions from measurements. Without additional tools, measurements cannot predict what would happen after a modification of the network or of its load.

Simulations do not scale and always rely on fairly arbitrary models. Often, simulation packages provide an illusion of accuracy and sophistication. Usually, they provided detailed results based on arbitrary assumptions. Sometimes, the simulations provide confidence intervals around estimated values. However, these confidence intervals assume that the assumptions are exact. Of course, simulations are very useful, necessary in fact, to verify that protocols are correct and to discover how to improve systems. Using simulations (e.g., [3]), you can verify that TCP is biased in favor of connections with a shorter round-trip time and that RED (Random Early Detection) and ECN (Explicit Congestion Notification, see [2]) help correct that bias and improve the utilization of the network links. You can also verify that WFQ (Weighted Fair Queuing, see [1]) helps protect classes of traffic against each other. There are many examples of useful applications of simulations in network engineering. However, simulations do not scale to a reasonable size. For instance, NS (see [4]), the network simulator developed at the Lawrence Berkeley National Laboratory, is one of the best simulation tool for TCP/IP networks. However, try using NS to simulate 100 TCP connections that go through a network with three nodes. You will discover that simulations are painfully slow. Moreover, the simulation results are difficult to trust. You would need to make a huge number of simulation experiments while varying the assumptions to get a sense of likely performance measures. The same comments can be made about commercial packages that usually focus more on giving the user a nice graphical interface than on simulation efficiency. See [5] for a discussion of the limitations of simulations of the Internet.

Analysis has had a disappointingly modest impact on network design. Although an impressive body of work is available on network performance evaluation, the fraction of that work that has influenced network engineering is very minute. This lack of impact is explained by the limitations of analysis. Exact analysis is tractable only for overly simplified models; approximate analysis provides results that may be difficult to trust. Most often, analysis provides an important qualitative insight and explains critical aspects of network operations. Although this insight can guide the development of new protocols, it is not sufficient to size a network and help its design.

Thus, network engineering tools are sorely needed but existing tools are inadequate. How can more useful tools be developed? Useful tools should have the following characteristics:

1. **Robust:** The tools should provide trustworthy bounds on the performance measures. These bounds should be valid for a wide range of modeling uncertainties.
2. **Fusion:** The tools should be able to incorporate results from measurements and from simulation experiments.
3. **Pro-Active:** The tools must be useful for network design and improvements, not only to analyze an existing network.

These basic requirements should serve as guidelines on how the tools should be designed.

The first requirement implies that the objective is not a detailed performance evaluation for tightly defined models of users, applications, and protocols. Rather, the objective is to identify the likely range of performance measures that correspond to a set of plausible models. If the set of models can be parameterized in a way that makes the analysis tractable for all the parameters, the range of performance measures can be determined. However, this is rarely the case and the changes typically correspond to different classes of models that must be analyzed separately. In any case, the lesson is that detailed results are not important. Instead, robust bounds on performance measures are more useful.

The second requirement means that the tools must accept descriptors that can be produced by measurements or simulation experiments. These descriptors might be bounds on some performance measures as a function of an operating point. That is, the tools should be adapted to the observations that can be made with measurements and simulations.

The third requirement implies that the tools must be able to examine different designs, possibly to perform some optimization. The tools should be flexible enough to be able to explore different technologies.

The tools themselves should probably involve a combination of analytical, numerical, and simulation methodolo-

gies. Brute force simulations are almost certainly hopeless to meet the requirements that were identified.

In this paper, we examine one simplified approach to evaluate the performance of TCP connections in a network. Our approach is motivated by the above requirements. The approach that we propose is a transaction-level model of the TCP connections. The model enables approximate analytical evaluations of the performance of the connections. Also, the model can include results from measurements and simulations about the dynamics of these connections.

2 TCP Performance

In this section we describe the network-engineering problem that we want to address. The basic problem is that of predicting the performance of TCP connections in a network. This problem is important because TCP-based applications generate about 85% of the Internet traffic. The problem is difficult because of the complex feedback behavior of TCP. We start with discussion of the TCP applications. We then propose a model of the network and of its operating conditions.

2.1 TCP applications

TCP applications consist in random requests for file transfers. At any given time, a random number of connections share a given link. The file transfer time is therefore a random variable. The network should be designed so that these transfer times are acceptably small. Equivalently, the network should deliver the files at a rate that is commensurate with the expectations of the users.

A detailed model of the connection process would capture the dependence of that process on the transmission rate of the network. That is, if the network is slow, then connections take longer, which further increases network congestion. However, it can be argued that if the network is slow, then some users may become frustrated and reduce their web-browsing activities. In our first model, we simplify this complicated connection process. We consider that there are many users and that a fraction of them have TCP connections that are active. Further, we assume that the users are active independently of one another and that each user is active with a relatively small probability. Accordingly, the number of active users is well approximated by a Poisson random variable. These assumptions result in the following model.

At some typical time, along path p there are $N(p)$ connections, where $N(p)$ is approximately Poisson with mean and variance $\lambda(p)$.

Roughly,

$$N(p) = \lambda(p) \left[1 + \frac{Z(p)}{\sqrt{\lambda(p)}} \right]$$

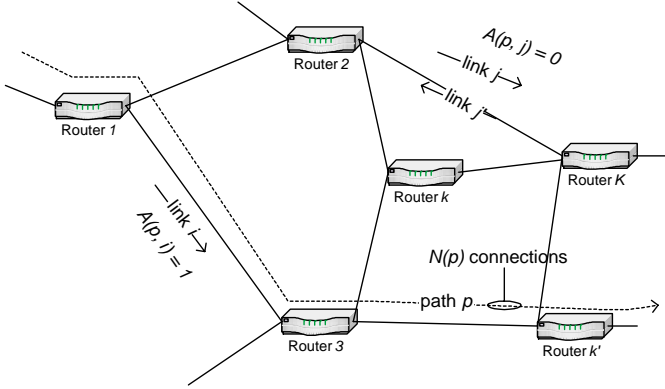


Figure 1. The network.

where $Z(p)$ is Gaussian with mean 0 and variance 1. In a first approximation, the random variables $Z(1), \dots, Z(p)$ are independent.

In a TCP model, one has to be explicit about the fact that the connections are full-duplex. In our first model, we neglect the traffic of ACKs. This approximation may be fine since the ACK traffic is typically less than 10% of the data traffic.

2.2 First Model of the Network

Our first model of the network consists of FCFS routers and TCP connections (see Figure 1). We know that TCP starts with a three-way handshake opening of the connection, followed by a slow-start phase until the source detects a loss, then TCP moves into a congestion-avoidance phase, and eventually the connection terminates with one two-way handshake for each half of the connection. The interactions of TCP connections in routers are complicated. If the router is drop-tail, it drops packets when it runs out of buffer space. If it is RED, then the router drops packets randomly with a probability that depends on the average queue length.

A detailed simulation can capture the intricate behavior of the routers and of the connections. These simulation experiments show that TCP attempts to achieve a fair-share of the network links. The simulations also show that TCP does not quite succeed. The models should reflect these simulation results. In a first model, we assume that TCP is ideal and shares the link bandwidths in a max-min fair way. In a later model, we examine how the performance measure change if we take into account the non-ideal behavior of TCP.

Path p goes through a set of links among $\{1, \dots, J\}$. Let A be the incidence matrix of the graph. That is,

$$A(p, j) = 1\{j \in p\}.$$

One can calculate the number $L(j)$ of connections along

link j as follows:

$$L(j) = \sum_p N(p)A(p, j).$$

We designate by $C(j)$ the transmission rate of link j . Note that the model is rich enough to capture the case where a given path carries connections from different groups of users that have distinct access rates. Indeed, to reflect this situation, one may consider that these different groups belong to different paths that go through the same links except for one additional link that models the limitation on the access rate.

In our discussion so far we have assumed that we know the average number of connections along the network paths. That is, we assume that $\lambda(p)$ is known for $p = 1, \dots, P$. How can the network designer estimate these mean values?

One possible scenario is as follows. The designer knows the topology of the network or proposes a topology. The number of users $U(r)$ attached to each router r (say ISP PoP) is estimated. The traffic pattern can be modeled by postulating that a fraction $R(r, r')$ of the connections that are generated by a host attached to router r are with a host attached to router r' .

The routing algorithm (BGP & OSPF) determines a mapping from the source/destination pair (r, r') to the path p . We denote this mapping by $p(r, r')$. A shortest path algorithm can determine this mapping.

Assume that each user has an active connection with probability α . Then, the average number $\lambda(p)$ of connections along path p is given by

$$\lambda(p) = \alpha \sum_{r, r'} U(r)R(r, r')1\{p(r, r') = p\}.$$

3 Analysis of First Model

One important information for improving the network is the identification of the network bottlenecks. A bottleneck is a link that limits the throughput of connections. More precisely, a link is a bottleneck if by increasing its transmission rate one can increase the rate of at least one connection. A useful refinement of that information is the number of connections that are limited by that bottleneck and by how much the bottleneck limits the throughput.

3.1 Min-max Equilibrium

We sketch a procedure for identifying the bottlenecks. In our first model, we assume that TCP achieves a min-max sharing of the links. For given numbers of connections along the paths, we compute the rate $R(p)$ per connection along path p . This number is obtained by performing the recursive calculation of the min-max equilibrium

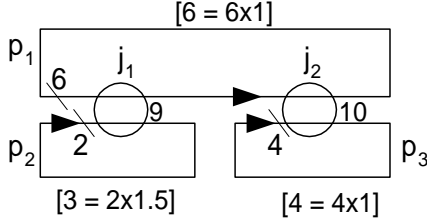


Figure 2. Network with Connections.

for the network with $N(p)$ connections along path p , for $p = 1, \dots, P$. We designate by

$$\mathbf{R} = \Phi(\mathbf{N}, \mathbf{C}) \quad (1)$$

the min-max vector of rates $\mathbf{R} = (R(1), \dots, R(P))$ that corresponds to the vector of numbers of connections $\mathbf{N} = (N(1), \dots, N(P))$ and to the vector of link rates $\mathbf{C} = (C(1), \dots, C(P))$.

To clarify our definition of Φ , consider the example shown in Figure 2. The figure shows two links (j_1 and j_2) of the network with rates $C(j_1) = 9$ and $C(j_2) = 10$, respectively. The network has other links that are not shown but that are not bottlenecks. The figure also shows three paths (p_1, p_2 , and p_3) that are used by $N(p_1) = 6, N(p_2) = 2$, and $N(p_3) = 4$ connections, respectively.

In general, a vector of rates (x_1, \dots, x_M) for connections is a max-min equilibrium if it is feasible for the network and if it is not possible to increase the rate x_m of one connection without decreasing some other rate that is smaller than or equal to x_m . The max-min equilibrium for this network corresponds to the following rates per path:

$$R(p_1) = 6, R(p_2) = 3, R(p_3) = 4.$$

Indeed, 8 connections share link j_1 with rate $C(j_1) = 9$, so that each connection could get a throughput of $9/8$ if there were no other bottleneck. Similarly, 10 connections share link j_2 with rate $C(j_2) = 10$, so that each connection could get a throughput of 1 if there were no other bottleneck. The max-min allocates a rate of 1 to each connection of paths p_1 and p_3 and shares equally the spare capacity of link j_1 (equal to $9 - 6 = 3$) among the connections of path p_2 .

3.2 First Bottleneck

Assume that all the connections should be treated equally. That is, there is no reason to give a preferential treatment to some connections over others.

Note that the min-max equilibrium is easy to identify. First compute, for each link, the ratio of the link rate divided by the number of connections that go through it. The minimum of these ratios is achieved by a link that we call the

“first bottleneck”. That ratio determines the rate per connection that goes through the first bottleneck. We call this rate the “minimum rate.” Allocate that minimum rate to the connections that go through the first bottleneck and subtract the rate of these connections from the capacity of the other links that carry some of these connections. Repeat the procedure for the network without the first link. Continuing in this way determines the equilibrium.

Because the numbers of connections are random, so is the link that is the first bottleneck and so is the minimum rate. We can determine the probability that a given link is the first bottleneck. For instance, in the case of the network of Figure 2, link j_1 is the first bottleneck if

$$\frac{C(j_1)}{N(p_1) + N(p_2)} < \frac{C(j_2)}{N(p_1) + N(p_3)}, \quad (2)$$

i.e., if

$$N(p_1)[C(j_1) - C(j_2)] < N(p_2)C(j_2) - N(p_3)C(j_1). \quad (3)$$

A tool can be built to evaluate this probability, for instance by approximating each $N(p)$ by a Gaussian random variable with mean and variance $\lambda(p)$. The tool can then determine the probabilities $\{\pi(j), j = 1, \dots, J\}$ where $\pi(j)$ is the probability that link j is the first bottleneck. The network designer can then increase the capacity of the most likely first bottleneck.

Specifically, in the case of the network of Figure 2, the inequality (2) is equivalent to $P(Z > 0)$ where Z is a Gaussian random variable with mean

$$10\lambda(p_1) + \lambda(p_2) - 9\lambda(p_3)$$

and variance

$$100\lambda(p_1) + \lambda(p_2) + 81\lambda(p_3).$$

For a concrete example, assume that $\lambda(p_1) = 30, \lambda(p_2) = 40$, and $\lambda(p_3) = 44$. We find that (3) becomes

$$\begin{aligned} P(N(-56, 6604) > 0) &= P(N(0.69, 1) < 0) \\ &= P(N(0, 1) < -0.69) \approx 0.25. \end{aligned}$$

That is, for this example, the bottleneck is j_1 with probability 25% and it is j_2 otherwise.

One can equalize the probability that each link is a bottleneck as follows. From (3) we see that if

$$\lambda(p_1)[C(j_1) - C(j_2)] = \lambda(p_2)C(j_2) - \lambda(p_3)C(j_3),$$

then each link is a bottleneck with equal probabilities. However, the case of a general network is more complex.

3.3 Rate per Connection

One useful measure of performance is the rate per TCP connection. A numerical tool can evaluate this rate as follows. Recall from (1) that the rate is some function of the numbers of connections. If we know the distribution of the number of connections, we can determine, in principle, the distribution of the rate per connection. We illustrate the calculations that the tool must perform for the network of Figure 2. To simplify the notation, let

$$\begin{aligned} x_i &= N(p_i) \text{ and } R_i = R(p_i), i = 1, 2, 3; \\ C_k &= C(j_k), k = 1, 2. \end{aligned}$$

With this notation, the identity (1) can be written as

$$\begin{aligned} R_1 &= \min\left\{\frac{C_1}{x_1 + x_2}, \frac{C_2}{x_1 + x_3}\right\} \\ R_2 &= \frac{C_1 - x_1 R_1}{x_2} \\ R_3 &= \frac{C_2 - x_1 R_1}{x_3}. \end{aligned}$$

A numerical integration tool can determine the probability that these rates are acceptably large. For instance, one design objective might be that these rates should all be at least equal to 0.1 (say 0.1Mbps) with a high probability. For our numerical example, we must evaluate the following probability:

$$P(x_1 + x_2 \leq 90 \text{ and } x_1 + x_3 \leq 100).$$

Since $x_1 + x_2$ and $x_1 + x_3$ are associated random variables, this probability is larger than

$$\begin{aligned} P(x_1 + x_2 \leq 90)P(x_1 + x_3 \leq 100) \\ = P(N(70, 70) \leq 90)P(N(74, 74) \leq 100) \\ = P(N(0, 1) \leq 2.4)P(N(0, 1) \leq 3) \approx 0.99 \end{aligned}$$

which may be an acceptable design objective.

A numerical tool could perform the calculation without using the lower bound. One possible method for doing this calculation for a large network is by a Monte Carlo simulation. In such a simulation, one first generates a random vector \mathbf{x} of the numbers of connections along the different paths. One then checks whether the inequalities are satisfied. The number of random variables to generate is equal to the number of paths and the number of inequalities to verify is equal to the number of links. One can estimate the number of random vectors to generate as follows. For a given random vector \mathbf{x}_n , let $f(\mathbf{x}_n)$ be equal to 0 if the inequalities are satisfied and to 1 otherwise. We want to estimate $Ef(\mathbf{x}_n)$ by computing

$$\frac{f(\mathbf{x}_1) + \dots + f(\mathbf{x}_n)}{n}.$$

Standard arguments show that to be within a few percent of the correct value with probability 95% one has to generate approximately $n = 200/Ef(\mathbf{x}_n)$ trials. For $Ef(\mathbf{x}_n) \approx 1\%$, one has to generate approximately 20,000 trials. The complexity of the simulation is linear in the product of the number of paths times the number of links.

4 Model Uncertainties

As we stated at the outset, to be useful a tool must be able to examine the effect of model uncertainties. In our prototype, we examine one source of uncertainty: the departure from fairness of TCP.

4.1 Note on TCP Bias

Simulations and analysis show that TCP is biased in favor of connections with a shorter round-trip time. For instance, if persistent connections share a single bottleneck, then the throughput of a connection is roughly inversely proportional to its propagation time. The bias is reduced when the routers implement RED (random early drop) and also when the network implements ECN (explicit congestion notification). Nevertheless, the bias remains significant. Versions of TCP that almost eliminate this bias have been developed. One of these versions is TCP-Vegas. Another method to reduce the bias is to use a form of per flow queuing in the routers. These mechanisms are rarely implemented.

The bias of TCP is more difficult to predict in a network with many bottlenecks. Simulations show that a reasonable estimate of the bias still corresponds to a throughput inversely proportional to the round-trip time.

The ratio of propagation times of connections that share a common router can be rather large. The maximum value of this ratio could be estimated by examining the network topology.

4.2 Analysis of Effect of TCP Bias

For the purpose of our study, we assume that the ratio of propagation times is bounded by 10. This range of propagation times translates into departures from the fair equilibrium. For instance, in the network of Figure 2, assume that the propagation time of path p_1 is ten times larger than that of paths p_2 and p_3 . To study the effect of these propagation times, first assume that link j_1 is the bottleneck. In that case, connections along path p_1 get a rate R_1 that is ten times smaller than the rate R_2 of the connections along path p_2 . Consequently,

$$9 = 6R_1 + 2R_2 = 6R_1 + 20R_1 = 26R_1,$$

so that $R_1 \approx 0.35$. Second, assume that link j_2 is the bottleneck. The corresponding constraint becomes

$$10 = 6R_1 + 4R_3 = 6R_1 + 40R_1 = 46R_1,$$

so that $R_1 \approx 0.22$. We conclude that link j_2 is the bottleneck and the rates are

$$R_1 \approx 0.22, R_2 = \frac{9 - 6R_1}{2} \approx 3.8, R_3 = \frac{10 - 6R_1}{4} \approx 2.2.$$

More generally, this procedure can be applied to determine the rates that correspond to a given maximum value of the ratio of propagation times. The tool can include these calculations and take them into account when computing the probability that all the connections get a specific minimum rate. We illustrate this calculation in the case of the network of Figure 2 where we assume that the numbers of connections are random as in Section 3.2. We want to compute a rate α so that the probability that the connections get at least a rate equal to α is at least 95%. By combining the method of Section 3.2 and the ratio of 10 for the propagation times, we see that we must calculate the following probability:

$$P\left(\frac{9}{x_1 + 10x_2} \geq \alpha \text{ and } \frac{10}{x_1 + 10x_3} \geq \alpha\right).$$

This probability is the same as the following one:

$$P(x_1 + 10x_2 \leq 9/\alpha \text{ and } x_1 + 10x_3 \leq 10/\alpha).$$

A lower bound on this probability is

$$\begin{aligned} &P(x_1 + 10x_2 \leq \frac{9}{\alpha})P(x_1 + 10x_3 \leq \frac{10}{\alpha}) \\ &= P(N(430, 4030) \leq \frac{9}{\alpha})P(N(470, 4430) \leq \frac{10}{\alpha}) \\ &\approx P(N(0, 1) \leq 0.14\beta - 6.8)P(N(0, 1) \leq 0.15\beta - 7) \end{aligned}$$

where $\beta = \alpha^{-1}$. We find that $\beta = 110$ guarantees that this lower bound is at least 95%. Consequently, the guaranteed bandwidth per connection is $\alpha = 1/110 = 9 \times 10^{-3}$. Recall that if TCP was not biased we could guarantee a rate 0.1 to each connection. This simple example shows that it is essential for the tool to be able to take such a bias into account.

5 Classes of Service

In this section, we explore how we can predict the effect of classes of service. We first explain what we mean by class of service. We then discuss an example of analysis.

5.1 CoS Implementation

When implementing class of service (CoS), a router classifies packets based on the type of service field of the IP

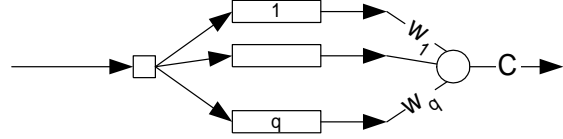


Figure 3. A router with CoS.

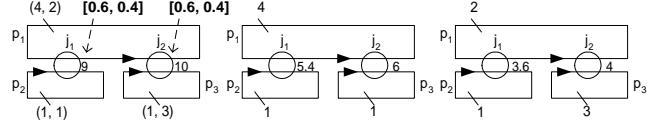


Figure 4. A network with CoS (left) and the network seen by each class.

header. The packets of different classes are sent to separate queues that the router serves according to a policy such as priority or weighted fair queuing or some variation. In this paper, we assume that the router implements a form of weighted fair queuing (WFQ). Figure 3 illustrates the queues in the router. The objective of CoS is to guarantee a minimum level of service to a class, independently of the load that other classes impose. This level of service is specified by *weights* in the case of WFQ. In the router of Figure 3, the packets are classified into q queues. Each queue k ($k = 1, \dots, q$) is allocated a weight w_k . These weights are nonnegative and add up to one. The meaning of the weights is that when all the queues are nonempty, the router allocates a fraction w_k of the link capacity to queue k . When only a subset of the queues are nonempty, the capacity of the link is allocated to the queues in proportion to their weight.

5.2 Network Example

The left-hand part of Figure 4 shows the same network as in Figure 2, except that it now implements CoS. There are two classes of service along each path. For instance, along path p_1 , there are 4 connections of class 1 and 2 connections of class 2, as indicated by the pair (4, 2). The weights in the routers are 0.6 for class 1 and 0.4 for class 2.

Let $R_{i,c}$ denote the rate of a connection of class c along path p_i (for $c = 1, 2$ and $i = 1, 2, 3$). Assuming that TCP achieves a min-max equilibrium, we find that, for this network, the connection rates are obtained one class at a time. That is, for class 1, we have to find the max-min equilibrium that corresponds to the middle part of Figure 4. For class 2, we have to find the max-min equilibrium for the network in the right-hand part of the figure.

Solving for the equilibrium values as we did previously,

we find the following rates:

$$R_{1,1} = 1.08, R_{2,1} = 1.08, R_{3,1} = 1.68$$
$$R_{1,2} = 0.8, R_{2,2} = 2, R_{3,2} = 0.8.$$

Comparing these rates with those shown in Figure 2, we can see the effect of introducing the class of service in the network. You will note that this effect is not straightforward and that suitable tools are needed to quantify it.

Note that in general, the determination of the rates of the connections cannot be done simply one class at a time. For instance, if one class does not saturate the rate that the router guarantees to it, then the unused rate becomes available to the other classes. However, the algorithm to compute the rates is a simple extension of the method we have described.

More generally, we see that the tool that we have described in the previous section can be used to evaluate the throughput of the different classes of service in a CoS network. The tool can also consider the bias of TCP and the randomness of the number of connections. This tool could be used to determine the suitable weights in the routers to achieve some target minimum rates for the different classes with a high probability. The analysis can be adapted easily to the case where the routers implement a priority scheduling combined with weighted fair queuing.

6 Conclusions

We argued that useful network engineering tools must produce robust results, be able to include results from measurements and simulations, and be helpful for design.

To illustrate these requirements, we proposed a prototype tool for predicting the performance of TCP connections in a network with or without CoS. This prototype uses statistics on user activity to model the number of connections along network paths. The prototype uses simulation results about TCP to quantify its bias.

The features of the tool that we propose include the following:

- The rates of TCP connections are not specified a priori by the sources. Rather, the rates are the result of the window-adjustment mechanism of TCP.
- The number of connections that use the network is random.
- The tool takes into account departures from fairness of TCP.
- The effect of CoS can be quantified.
- The tool can be used for network design.

To convert this prototype into a tool that can be used for guiding the evaluation or the design of networks, the algorithms and numerical procedures need to be implemented.

Acknowledgments

This research was performed while the author was on leave with Odyssea Systems, Inc. Special thanks to Rajarshi Gupta, Matthew Siler, Linhai He, and Eric Chi for their comments.

7 References

- [1] Floyd, S., and Jacobson, V., "Link-sharing and Resource Management Models for Packet Networks," *IEEE/ACM Transactions on Networking*, Vol. 3 No. 4, pp. 365-386, August 1995.
- [2] Ramakrishnan, K.K., and Floyd, S., "A Proposal to add Explicit Congestion Notification (ECN) to IP," *RFC 2481*, January 1999.
- [3] J. Mo, R. La, and J. Walrand, "Analysis and Comparison of Reno and Vegas," *INFOCOM'99*
- [4] <http://www-mash.cs.berkeley.edu/ns/>
- [5] Paxson, V., and Floyd, S., "Why We Don't Know How To Simulate The Internet," *Proceedings of the 1997 Winter Simulation Conference*, December 1997.