Network Performance for TCP Networks
Part I: Persistent Sources

Matthew Roughan[a], Ashok Erramilli[b] , and Darryl Veitch[c]

[a]AT&T Labs – Research, 180 Park Av., Florham Pk, NJ, 07932, USA
Email: roughan@research.att.com

[b]QNetworx Inc., Morganville, New Jersey, USA
Email: ashok@qnetworx.com

[c]Ericsson EMULab, Department of Electrical Engineering,
University of Melbourne, Victoria 3010, AUSTRALIA.
Email: m.roughan@ee.mu.oz.au

A method is presented which can form the basis for capacity planning for TCP traffic from persistent sources. The method estimates the rates of flows across an arbitrary network as determined by the TCP flow control, and can therefore estimate the performance of a TCP network. The method is simple, fast and provides good 1st order estimates with significant robustness. The method is extendible to a-persistent sources.

1. Introduction

The majority of traffic on the current Internet uses Transmission Controls Protocol (TCP) based transport. Engineering the access, core and backbone networks that constitute the Internet therefore requires the development of accurate analytic models of TCP performance. While analytic models employ a greater degree of abstraction than simulation models, they are better suited to support engineering tasks such as network design, capacity planning, and setting operational thresholds. These require the evaluation of a wide range of scenarios covering many parameter dimensions, which would imply excessive run times with a detailed simulation model.

A crucial component of the Transmission Controls Protocol (TCP) is the dynamic window flow control designed (by Van Jacobson) to prevent congestion collapses such as that which crippled the Internet in October 1986. Given the complexity of TCP window and timer dynamics, one might conclude that analytical models of TCP performance are infeasible. However, in recent years there have been a number of results using simplified analytic approaches that match those produced from highly detailed simulation models. For example, there have been several interesting results presented relating the average window sizes in the flow control to loss rates at the bottleneck of a network, which are notable for their simplicity. In this paper, we build on these simple relationships to develop models to predict network performance, from first principles, (in much the same vein as [1], but with some significant differences). The methodology developed here can be used to solve a number of engineering problems, such as estimating the capacity required in the network to meet certain performance objectives, for example providing a minimum throughput per source. TCP traffic can be considered to be *elastic* in the sense that it will conform to fill the available capacity – however in a complex network it may not do this in a fair manner [2,3]. For instance, it has been shown that TCP connections with longer Round Trip Times (RTT)'s receive a smaller proportion of the available capacity, as do connections which traverse multiple bottlenecks. Therefore we require a method which will allow us (among other things) to predict how the available capacity will be shared amongst competing TCP sources.

In this paper we limit ourselves to persistent or 'greedy' sources – that is sources which always have data to send. This is unrealistic in that there is evidence that the majority of TCP connections (for instance WWW file downloads) involve relatively short transfers. However, sources involved in long file transfers still contribute a significant proportion of the bytes over a network, because of the heavy-tailed nature of file sizes [4–7]. Thus, transfers which are large enough to be treated as infinite (in the sense that the connection is active over the time scale of engineering interest), and which therefore correspond to greedy sources, are significant. In the future we plan to extend our approach to a-persistent sources. We furthermore assume that for the purposes of dimensioning the elastic traffic and real-time traffic are partitioned. To some extent the use of fair queue scheduling provides such a separation.

There are several features which we require of our performance methodology:

- **Simplicity:** The method should require only a small number of *available* inputs. Methods which are highly parameterized, or require inputs that cannot be observed or estimated within a network, cannot be applied in practice.
- **Robust accuracy:** A method must provide qualitatively correct solutions under a wide range of scenarios, and be insensitive to the exact parameters of the system.
- **Speed:** Optimizing network configurations involves investigating many scenarios, implying high computational needs (particularly with heuristics such as Simulated Annealing or Genetic Algorithms). Hence we require fast algorithms scalable to large networks.

The abstractions used in the analytic models described here mean that they can only be regarded as 1st order approximations to a full system. Given the rapid growth and evolution of the Internet however, this is not a disadvantage for many engineering applications, as traffic forecasts are only accurate to an order of magnitude. Only when networks stabilize will further refinements become practically important.

The method proposed here is simple, requiring only fundamental inputs such as network topology and link capacities, very fast (based on matrix algebra), and provides solutions which are qualitatively correct as compared to detailed simulations. Further work is required to fully assess robustness.

In Section 2 we describe the background of this work, the so called $1/\sqrt{p}$ law, and the link analysis of [8] based upon it. In Section 3 we describe our generalization to networks. In Section 4 the method is used to examine a set of network scenarios considered elsewhere, to show that

- we can reproduce their results,
- the results are consistent with simulations,

- the method scales to reasonably large networks,
- numerical stability, at least in the cases considered.

We briefly discuss extensions to account for queueing delays in Section 5 before concluding the paper in Section 6. Familiarity with TCP flow controls is assumed (consult [9–14]).

2. Background

2.1. Inverse square root p law

There are a number of publications [3,15–17] which have demonstrated that under suitable assumptions the average window size of a source goes as the inverse square root of the loss probability: $w = k/\sqrt{p}$. Although the assumptions in any one case are somewhat restrictive (random independent losses, persistent sources, low loss rate,...) the result holds for a range of different conditions (varying from independent losses to periodic losses, through different versions of TCP (Tahoe, Reno, Delayed Acks, ...)) and non-negligible loss rates, and in fact seems to apply to persistent traffic on real networks, with $k \approx 1$. Furthermore, as Random Early Detection (RED) [18] and Selective ACKnowledgement (SACK) (Selective Acknowledgment) TCP become more common, the assumptions behind the law should be more accurate. We refer to this as the *inverse square root p law* or simply *law*.

There are also more complex formula that apply over a wider range of values of p [17], or for more complex loss patterns [20]. These laws all seem to be of the form

$$w = f(p), \tag{1}$$

where f is a non-increasing function. That is, as the loss-rate increases the average window size decreases (or possible remains constant). We shall only consider the simple square root law here, but note that the results can be easily extended to the more general formula as needed (for instance using the formula of [17]).

2.2. Link performance

The law summarizes many aspects of the dynamics, describing the performance of TCP file transfer via a fixed relationship between window size w and a parameter, the loss rate p. However the value of p is not specified, so the law alone cannot predict network performance. This lack is addressed in [8], where p is taken to be a function of w, allowing the average window size $\overline{w}$ of a set of n TCP sources sharing a resource of capacity C to be calculated. Rewriting the law with $k = 1/\sqrt{2}$ and $p = p(\overline{w})$ we have

$$\frac{\overline{w}^2}{2} p(\overline{w}) - 1 = 0. \tag{2}$$

To determine p we need a new, independent relationship, obtained by noting that the loss rate (under zero buffer assumptions) is just the proportional excess of the send rate $n\overline{w}$ over the capacity ($[\cdot]^+$ takes the positive part)

$$p(\overline{w}) = \frac{[n\overline{w} - C]^+}{n\overline{w}}, \tag{3}$$

where time is in units of RTTs, w in packets, and C in packets per RTT. Combining (2) and (3) we can solve for $\overline{w}$ by noting that $n\overline{w} > C$, so that $[\cdot]^+$ does not bite, thus

$$\overline{w} = \frac{1}{2}\left[C/n + \sqrt{(C/n)^2 + 8}\right]. \tag{4}$$

This is a simple first attempt at providing estimates of window sizes, and therefore throughputs, from first principles. To analyse a network we must take into account multiple routes, with different RTTs and loss probabilities.

3. Network Performance

In our extension to a network we add a key new assumption: that losses are independent of each other, so that route losses can easily be calculated from link losses. The approach is similar to that of the Erlang fixed point methods described in [21], however rather than applying the reduced load approximations leading to Erlang blocking type probabilities, relations (2) and (3) are extended in the most direct way possible, and we seek their resolution in a vectorial fixed point analogue of (4).

Boldface upper (lower) case are used for matrices (vectors), and $\mathbf{1}$ is a vector of 1's of the appropriate length. In addition, for some vectors $\mathbf{c}$ we define a corresponding matrix $\mathbf{C} = \mathrm{diag}(\mathbf{c})$ whose diagonal entries take the elements of $\mathbf{c}$, else are zero. We use MATLAB's notation $.*$, $./$ etc. to denote elementwise operations, and logarithms of matrices and vectors are likewise elementwise.

Consider a network with N links with capacities $\mathbf{c} = (c_j)$ and propagation delays $\mathbf{d} = (d_j)$. The R routes active in the network are defined by the link subsets $r_i \subset \{1, \dots, N\}$, and summarized in the $R \times N$ routing matrix $\mathbf{A} = [a_{ij}]$:

$$a_{ij} = \begin{cases} 1, & \text{if } j \in r_i, \\ 0, & \text{if } j \notin r_i. \end{cases}$$

Other route parameters are the Maximum Segment Size (MSS) $\mathbf{m} = (m_i)$ and the number of (entirely equivalent) greedy connections $\mathbf{n} = (n_i)$ sharing route i. All of the above parameters follow from a given network topology, an example is given in Section 4.

If we assume a simple symmetric model where the forward path for packets is the same as the return path for acknowledgments, then we can define the network by duplex links, (halving R) and so using the zero buffer assumption we can take the RTT $\mathbf{t} = (t_i)$ for each connection on route i as twice the contributing link propagation delays:

$$\mathbf{t} \;=\; 2\mathbf{Ad}. \tag{5}$$

MSS-dependent packet transmission times could easily be included, but the effect would be marginal compared to queueing, and is omitted here. Extensions to include queueing delays are possible, as discussed in Section 5. For networks with a high bandwidth-delay product both these terms are small. The effect of delayed acknowledgments could be included by adding a constant to $\mathbf{t}$.

Defining a per-route average window size vector $\mathbf{w}$, the law (2) immediately generalizes to $w_i^2 p_i(\mathbf{w})/2 - 1 = 0$, or

$$\frac{\mathbf{W}^2}{2}\mathbf{p}(\mathbf{w}) - \mathbf{1} \;=\; \mathbf{0}, \tag{6}$$

that is the law is assumed to hold separately for each route. The independent loss assumption gives route losses as a function of link losses $\mathbf{q}$ through

$$\mathbf{p}(\mathbf{w}) \;=\; \mathbf{1} - e^{\mathbf{A}\ln(\mathbf{1}-\mathbf{q}(\mathbf{w}))}, \tag{7}$$

and the link losses are given by some function

$$\mathbf{q} = \mathbf{g}(\mathbf{b}; \mathbf{c}), \tag{8}$$

$\mathbf{b}$ being the offered link traffics ($\mathbf{b}, \mathbf{c}$ in bits/s) given by

$$\mathbf{b} \;=\; \mathbf{A}^T \mathbf{s}, \tag{9}$$

where

$$\mathbf{s} = \mathbf{w}.*\mathbf{n}.*\mathbf{m}./\mathbf{t} \tag{10}$$

is the total route send rates. For the moment we shall once again approximate the loss function by the simple excess:

$$\mathbf{q} = \frac{[\mathbf{b}-\mathbf{c}]^+}{\mathbf{b}} = \left[\mathbf{1} - \frac{\mathbf{c}}{\mathbf{b}}\right]^+, \tag{11}$$

The problem consists of simultaneously solving the non-linear vector equations (6) and (7) for a fixed point $\mathbf{w}^*$, that is of finding a root of $\mathbf{F}(\mathbf{w}) = \frac{\mathbf{w}^2}{2}\mathbf{p}(\mathbf{w}) - \mathbf{1}$. As before, we deal only in deterministic smooth average traffic in this approach, loss being generated by excessive offered load. The law (6) implies that at $\mathbf{w}^*$ all routes must suffer loss.

3.1. Solving the equations

The Newton-Raphson method [22, Chp 9.6-9.7] is a standard way of finding a root of a set of vector equations such as the $\mathbf{F}(\mathbf{w}) = \mathbf{0}$ above. From an initial estimate $\mathbf{w}_0$, iteration is performed according to $\mathbf{w}_{n+1} = \mathbf{w}_n + \delta\mathbf{w}$, where the $\delta\mathbf{w}$ are obtained from the equation

$$\mathbf{J}\,\delta\mathbf{w} = -\mathbf{F}, \tag{12}$$

$\mathbf{J}$ being the Jacobian matrix of $\mathbf{F}(\mathbf{w})$, until F is sufficiently close to zero.

For the network fixed point problem as defined above the Jacobian can be calculated analytically, and is

$$J_{ij} = \frac{\partial F_i}{\partial w_j} = \begin{cases} w_i p_i + \dfrac{w_i^2}{2}\dfrac{\partial p_i}{\partial w_j}, & i = j, \\ \dfrac{w_i^2}{2}\dfrac{\partial p_i}{\partial w_j}, & i \neq j, \end{cases} \tag{13}$$

where

$$\frac{\partial p_i}{\partial w_j} \;=\; (1-p_i)\sum_{k=1}^{N} a_{ik} y_{kj} \tag{14}$$

and

$$y_{kj} \;=\; \begin{cases} \left[\frac{a_{jk}\, n_j m_j\, t_j^{-1}}{b_k}\right], & q_k > 0 \\ 0, & q_k \leq 0. \end{cases} \tag{15}$$

In this case we can calculate the derivatives analytically, but it is only slightly more computationally complex to numerically evaluate the derivatives, via multiple evaluations of $\mathbf{F}$ [22, Chp 5.7]. Numerical differentiation can be used if a more complicated loss function $\mathbf{g}(\cdot)$ or window size law $\mathbf{f}(\cdot)$ is used in the problem formulation.

3.2. Existence, convergence, initial conditions

An alternative viewpoint is to consider the R dimensional dynamical system formed by $\mathbf{w} \leftarrow f(\mathbf{w}) = k/\sqrt{p_r(\mathbf{w})}$. The solution above is also the fixed point $\mathbf{w}^*$ of the dynamical system, satisfying $\mathbf{w}^* = f(\mathbf{w}^*)$. If the functions $\mathbf{f}$ are continuous, as they are in the inverse square root case, then Brouwer's fixed-point theorem [23,24] can be used to conclude that at least one solution $\mathbf{w}^*$ exists (the second condition for Brouwer's theorem, that the functions act on a compact set, can also be satisfied). The question of uniqueness is more difficult and requires further work, however, it would seem that any reasonable choice of the $\mathbf{f}$ will be non-increasing. This monotonicity greatly constrains the possible dynamics. Iterating the dynamical system can now be seen naturally as a Repeated Substitution (RS) method for searching for the fixed point, and a necessary condition for convergence is local stability about $\mathbf{w}^*$. It is not difficult to see (for example in one dimensional example), that local instability is possible and therefore that a blind RS approach may fail. However, by damping, convergence can be guaranteed (at the cost of more iterations).

Also, as in many such situations, the choice of initial conditions may play a part in both eventual convergence, and number of iterations require for convergence. We have tried a number of *ad hoc* approaches for choosing initial conditions, all of which perform well with Newton-Raphson. One must be much more careful with repeated substitution, as a poor choice can result in divergence, or very slow convergence.

In all the cases presented below, and many others, a direct method using an initial condition corresponding to twice the window size giving the onset of loss on the probable worst bottleneck, found a highly robust physical solution ($p_i > 0$ for each route) in around 12 iterations. Importantly, the number of iterations to convergence, using a fixed absolute criterion of 10^{-6}, was found to be almost independent of the size of the network, being no more than 15 even for large networks.

3.3. Discussion

One of the key justifications of our approach is the generality of the inverse square root p law, where changes in underlying assumptions are largely accommodated by a simple change in k. Our approach can be be trivially adapted to arbitrary scalar (or vector) k, could be used with other laws altogether, with loss formula more complex than simple excess, and can also be made to include further details of TCP dynamics. For instance TCP receive windows could be included by having access links with zero propagation delay and a bandwidth determined by the maximum window size. Alternatively, they could be included directly by using the more general law given in [17].

4. Numerical Example

4.1. Multiple congestion points

We consider the multiple congested gateways network class of [2]. *Figure 1* shows the network with the number of bottlenecks $n = 2$. The thick links are 10 Mbps links with propagation delay ε ms, while the thin links are 1.5 Mbps links with propagation delay δ ms (as in [16]). For simplicity assume 1 connection per route, and 1000 byte packets. The full configuration is

$$\mathbf{c}^T = (10, 1.5, 1.5, 1.5, 10, 10, 10, 10, 10) \times 10^6, \tag{16}$$

$$
\begin{aligned}
\mathbf{d}^T &= (\varepsilon,\delta,\delta,\delta,\varepsilon,\varepsilon,\varepsilon,\varepsilon,\varepsilon), &(17)\\
\mathbf{A} &= \begin{pmatrix} 1&1&1&1&1&0&0&0&0\\ 0&1&0&0&0&1&1&0&0\\ 0&0&0&1&0&0&0&1&1 \end{pmatrix}, &(18)\\
\mathbf{n}^T &= (1,1,1), &(19)\\
\mathbf{m}^T &= (8000,8000,8000). &(20)
\end{aligned}
$$

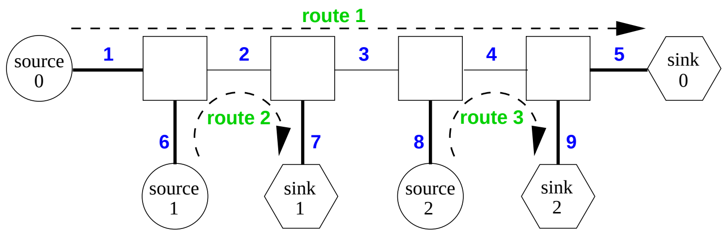

Figure 1. Example network with 2 bottlenecks (links 2 & 4). Sources, sinks and routers are shown as circles, hexagons and squares respectively, dashed lines show routes. The thicker lines on access links indicate their higher bandwidth and lower delay.

A simplified system is obtained by assuming that losses occur only on the expected bottlenecks at (links 2 & 4 in the figure), whose (equal) loss rates thereby completely determine the system. The components of $\mathbf{w}^*$ would then be

$$
w_1 = \frac{1}{2}\left[\frac{c}{l} + \sqrt{\left(\frac{c}{l}\right)^2 + \frac{8}{n}}\right], \tag{21}
$$

and $w_2 = \cdots = w_{n+1} = \sqrt{n}w_1$, where $l = 1/t_1 + \sqrt{n}/t_2$. This explicit solution is a useful check on the Newton-Raphson $\mathbf{w}^*$ of the full system, and gives the true fixed point when the gateway links are indeed the only congested ones. In the cases tested the full method converged to (21), consistent with the results in [2,16].

We tested the method against simulation using `ns` [25] version 2.1b5, using RED buffers with the default `ns` RED parameters, and 100 packet buffers. Two important facts emerge from the simulation. First, the utilization of the first bottleneck (of n) is around 90%, reflecting the imperfect efficiency of TCP connections. From this point of view the buffers are too small, producing starvation on the link. Second, the RTTs are notably above the propagation delays (more than double in cases), indicating that the queues are not small compared to the bandwidth delay product. Thus, in these networks, the 100% bottleneck utilization and zero buffer idealizations of our model do not hold well. In more realistic networks, where buffers are larger, efficiencies will approach 100%, though at the possible cost of even more significant queueing delay as a fraction of RTT. Although order of magnitude agreement was obtained, we focus on presenting results incorporating heuristic corrections for efficiency and RTT. We argue that more natural solutions will emerge from future work. We use $k = \sqrt{2}$ here.

Figure 2 shows the measured simulation send rates (o) and simple predictions (□) of the send rates of routes 1 and 2 (for a network with n bottlenecks), with the correction to the ns-measured efficiency by reducing **c**. The differences are small, of the order of the (very small)[1] confidence intervals from the simulation. Note that the method correctly predicts the proportion of capacity that each source gets. These results are consistent with those in [2,16]. The figure also shows the send rates when the real measured RTT is used in the calculation, and when a queueing approximation (discussed in Section 5) is used. The differences between these alternate solutions are all small.

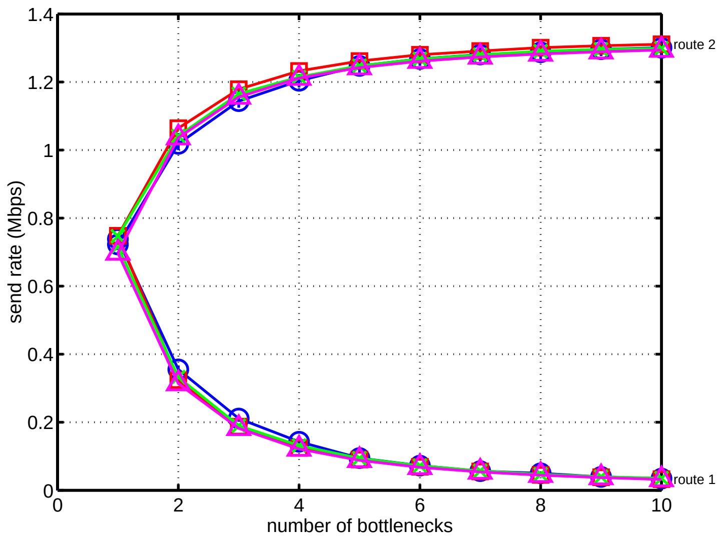

Figure 2. Send rates for routes 1 and 2. The symbols show: (o) the simulation results (including vertical bars showing 95% confidence intervals), (□) send rates with adjustment for efficiency, ($\times$) using measured RTTs, ($\triangle$) using the M/D/1/K queueing approximation for loss and RTT estimates.

Figure 3 shows the simulated (o) and predicted (□) average window sizes, also with the adjustment for efficiency. We see that the although the shape of the curves is correct, the values are quite different. This is because a second adjustment is needed, this time to the RTTs. The third curve (x) shows the results adjusted both in efficiency and using the ns-measured RTTs, and good agreement is found. A similar story is told in *Figure 4* for the corresponding loss probabilities. In the queueing results ($\triangle$) the RTT including queueing is estimated (discussed below), resulting in results very similar to those when the correct RTT is used in the prediction.

It is noteworthy that the relative send rate predictions are accurate even without accurate RTTs. This is likely to be due to the total send rate being constrained to be close to the link bandwidth, and the fact that the square root p law constrains the ratios of the relative send rates. A simple system such as this is particularly important because it tests the limits of the independent loss assumption – the losses here are clearly not

[1]The confidence intervals for the simulation results are shown as vertical bars about the (o), but are very small.

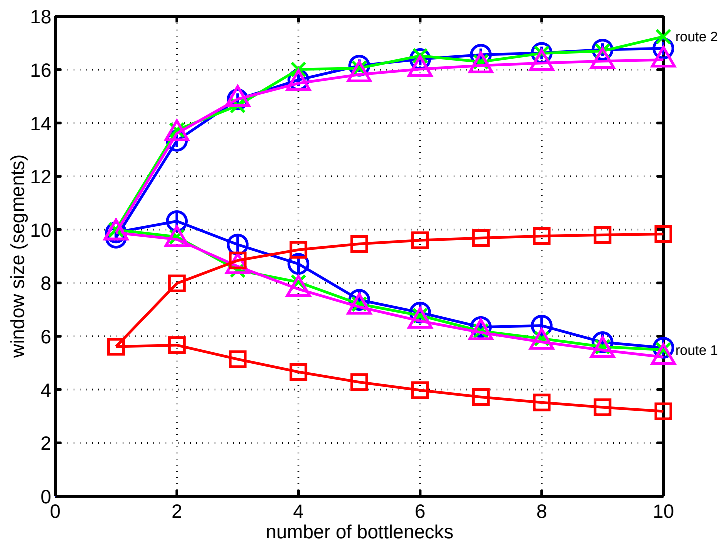

Figure 3. Window sizes for routes 1 and 2. The symbols show: (o) the simulation results (including vertical bars showing 95% confidence intervals), (□) send rates with adjustment for efficiency, (×) using measured RTTs, (△) using the M/D/1/K queueing approximation for loss and RTT estimates.

independent. However, the results show that, as one might expect for low loss rates, the violation of the independence assumption has little effect on the final results. Furthermore, by analogy to the Erlang-B results, it is likely that as the system becomes larger the independence assumption will hold more accurately.

In all cases, the algorithm converged in less than 12 iterations, with computation times well below 1 second[2]. Problems with other parameters (e.g. $n \neq 1$), and topologies have been solved, including some involving hundreds of nodes, all in under 15 iterations, although computations still increase with A.

5. Estimating Queueing Delays

There are several issues left unsolved by the previous work. Namely:

- The RTT estimates based simply on propagation delay are clearly inaccurate, and have a strong influence on the estimates of window sizes.
- The loss probability estimate is a crude approximation. It allows loss only when the load exceeds the systems capacity, whereas in reality loss can occur at lower loads due to the variability of the traffic. The net result of this approximation is that we must include an efficiency factor into our results.

Starting as simply as possible we can attempt to solve both problems by using a simple queueing model to calculate loss and delay in the system. The simplest viable model is the M/D/1/K queueing model. The assumptions of this model are that the arrivals are

[2]All computations and simulations were carried out on a Linux PC, with a 600 MHz Pentium III processor and 128 MB of RAM.

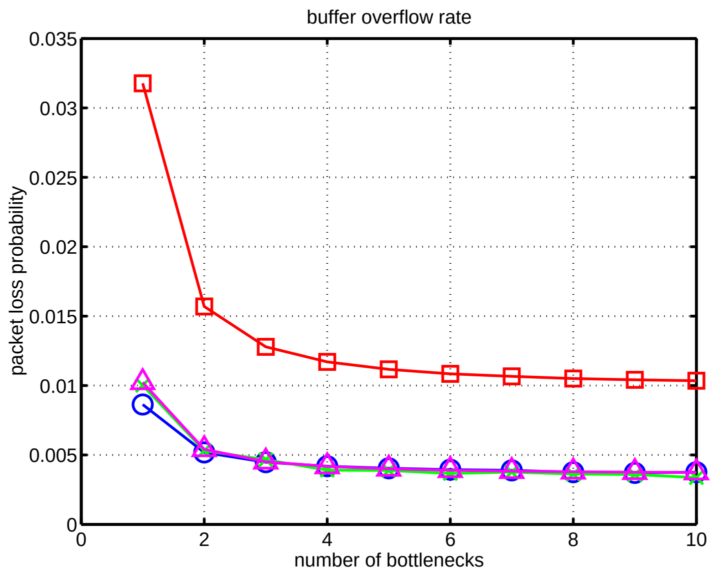

Figure 4. Loss rates for the first bottleneck. The symbols show: (o) the simulation results (including vertical bars showing 95% confidence intervals), (□) send rates with adjustment for efficiency, (×) using measured RTTs, (△) using the M/D/1/K queueing approximation for loss and RTT estimates.

Poisson, the service times (or packet sizes) are fixed, there is single server (the link) and there is space for $K-1$ packets in the buffer. Although the assumption of fixed packet sizes may not be realistic in practice, it matches the simulations above, and is far more realistic than the M/M/1 model which assumes exponential packet sizes. The Poisson input assumption is obviously flawed, as is the queueing model (as the simulations use RED queueing management), however, as we shall see below, it is a useful first approximation.

In this case we may calculate average delays and loss probabilities from [26, pp. 179-182]. The calculation allows loss at loads less than the system capacity, and therefore we no longer need an efficiency factor in the solution. We may also use the estimated queueing delays by first solving the system of equations with zero queueing, then estimating the queueing delays from the first order solution, and using these to obtain a second order solution. We might iterate this procedure until it converges, but in practice it seems that additional iterations provided dubious improvements. That is, some performance measures are slightly more accurate while others are slightly less. This seems to indicate that the M/D/1/K model is not perfectly matched to the actual queueing system.

One can observe, in Figures 2, 3, and 4 that the results are not noticeably better than those using correct RTTs and efficiencies, but the new results do not require any factors other than the network parameters.

As noted above, the simple M/D/1/K model does not seem to be perfectly matched to the real queueing system. We can further improve the results using Mean Value Analysis (MVA). In standard analysis, flow controlled networks are typically modeled by closed networks of queues in which the network population is equated to the window size (or the aggregate window size of all the flows). Such a model can be analyzed under restrictive product form assumptions using a variety of numerical techniques, for example, the well-known Mean Value Analysis (MVA). Several authors have argued that TCP has the effect of sharing available capacity, and in that it resembles processor sharing (PS), which would satisfy the product form assumption. Our experience is that standard analysis methods are not accurate because the window size of TCP connections varies dynamically. Secondly, TCP tends to clump traffic, in that IP packets often travel in batches. In our on-going work, we have used extensions to MVA (EMVA) that incorporate heuristic corrections to account for TCP clumping. The resulting fixed point equation can be solved by repeated substitution. Initial results are highly promising, though additional work is required on the robustness of such a procedure, and speeding up the computation.

6. Conclusion

This paper has addressed the problem of predicting TCP performance in networks from basic inputs such as network topology and link capacities. The solution is based on a 1st order estimate of the throughput of a link as a function of loss rate, the "inverse square root p law". Using this law, together with a loss equation in a network setting, a set of equations were derived that can be numerically solved using the Newton-Raphson method. The solution appears to be simple and relatively fast even for moderately large networks.

Current limitations of the method were highlighted, mainly due to the assumption of persistent sources, but also including the inaccuracies of the first order loss law, queueing models, and square root p law. However, we are still able to approximate the simulation results (from ns) with fair degree of accuracy. At this early stage the method shows much promise as a simple, rapid way to obtain 1st order solutions to important engineering problems for quite large networks.

There are many directions for future work including:

- Using more accurate queueing models – Extended Mean Value Analysis shows promise – to obtain more robust RTT estimates, and loss probabilities.
- Solving for transient (non-persistent) sources.
- How to incorporate TCP timer behavior in further refining estimates of RTT.

Acknowledgment

The majority of this work was conducted while Matthew Roughan was a member of the Ericsson EMULab, at the University of Melbourne. This work was made possible by support from Ericsson Research.

REFERENCES

1. R. Gibbens, S. Sargood, C. V. Eijl, F. Kelly, H. Azmoodeh, R. Macfadyen, and N. Macfadyen, "Fixed-point methods for the end-to-end performance analysis of IP networks," in *13th ITC Specialist Seminar on Internet Traffic Measurement and Modelling*, 2000.
2. S. Floyd, "Connections with multiple congested gateways in packet-switched networks, part I: One way traffic," *Computer Communications Review*, vol. 21, no. 5, 1991.
3. T. V. Lakshman and U. Madhow, "The performance of networks with high bandwidth-delay products and random loss," *IEEE/ACM Transactions on Networking*, 1997.

4. P. Barford and M. Crovella, “Measuring web performance in the wide area,” *Performance Evaluation Review: Special Issue on Network Traffic Management and Workload Characterisation*, 1999.
5. W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, “Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level,” *Proceedings of the ACM/SIGCOMM’95*, 1995.
6. M. E. Crovella and A. Bestavros, “Self-similarity in World Wide Web traffic: Evidence and possible causes,” *IEEE/ACM Transactions on Networking*, vol. 5, December 1997.
7. M. E. Crovella, M. Taqqu, and A. Bestravos, in *A Practical Guide to Heavy Tails: Statistical Techniques and Applications,* eds. Robert J. Adler and Raisa E. Feldman and Murad S. Taqqu, ch. Heavy-tailed probability distributions in the World Wide Web, pp. 3–25. Birkhäuser, Boston, 1998.
8. P. Key, “Fixed point methods and congestion pricing for TCP and related schemes,” in *Workshop on the modeling of TCP*, (École Normale Supérieure), 1998.
9. G. R. Wright and W. R. Stevens, *TCP/IP Illustrated, Volume 2.* Addison-Wesley Publishing Company, 1995.
10. V. Jacobson, “Congestion avoidance and control,” *Communication Review*, vol. 18, no. 4, pp. 314–329, 1988.
11. “Transmission Control Protocol.” IETF RFC 793, September 1981.
12. T. Socolofsky and C. Kale, “A TCP/IP tutorial.” IETF RFC 1180, 1991.
13. V. Jacobson, R. Braden, and D. Borman, “TCP Extensions for High Performance.” IETF RFC 1323, 1992.
14. M. Allman, V. Paxson, and W. Stevens, “TCP congestion control.” IETF Network Working Group RFC 2581, 1999.
15. T. J.Ott, J. Kemperman, and M. Mathis, “The stationary behavior of ideal tcp congestion avoidance.” Available at `ftp://ftp.bellcore.com/pub/tjo/TCPWindow.ps`, 1996.
16. M. Mathis, J. Semke, J. Mahdavi, and T. Ott, “The macroscopic behavior of the TCP congestion avoidance algorithm,” *Computer Communication Review*, vol. 27, pp. 67–82, July 1997.
17. J. Padhye, V. Firoin, D. Towsley, and J. Kurose, “Modeling TCP throughput: A simple model and its empirical validation,” in *ACM SIGCOMM’98*, 1998.
18. B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, “Recommendations on queue management and congestion avoidance in the internet.” IETF Network Working Group RFC 2309, April 1998.
19. K. Fall and S. Floyd, “Simulation-based comparisons of Tahoe, Reno, and SACK TCP,” *Computer Communication Review*, vol. 26, no. 3, pp. 5–21, 1996.
20. E. Altman, K. Avrachenkov, and C. Barakat, “A stochastic model of TCP/IP with stationary random losses,” in *SIGCOMM’2000*, 2000.
21. F. Kelly, “Blocking probabilities in large circuit switched networks,” *Advances in Applied Probability*, vol. 18, pp. 473–505, 1986.
22. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The art of Scientific Computing.* Cambridge University Press, 2nd ed., 1988-92.
23. L. Brouwer, “Beweis der invarianz der dimensionzahl,” *Math. Ann.*, vol. 70, pp. 161–165, 1911.
24. V. Istratescu, *Fixed Point Theory.* Reidel, 1981.
25. “UCB/LBNL/VINT network simulator - ns (version 2).” `http://www.isi.edu/nsnam/ns/`.
26. R. Cooper, *Introduction to Queueing Theory.* The Macmillian Company, 1972.