Differentiated Control of Web Traffic: A Numerical Analysis*

Liang Guo and Ibrahim Matta
Computer Science Department, Boston University
Boston, MA 02215, USA
{guol, matta}@cs.bu.edu

ABSTRACT

Internet measurements show that the size distribution of Web-based transactions is usually very skewed; a few large requests constitute most of the total traffic. Motivated by the advantages of scheduling algorithms which favor short jobs, we propose to perform differentiated control over Web-based transactions to give preferential service to short web requests. The control is realized through service semantics provided by Internet Traffic Managers, a Diffserv-like architecture. To evaluate the performance of such a control system, it is necessary to have a fast but accurate analytical method. To this end, we model the Internet as a time-shared system and propose a numerical approach which utilizes Kleinrock's conservation law to solve the model. The numerical results are shown to match well those obtained by packet-level simulation, which runs orders of magnitude slower than our numerical method.

Keywords: Heavy-tailed Distributions, TCP Congestion Control, Traffic Engineering.

1. INTRODUCTION

Previous job scheduling studies indicate that providing rapid response to interactive jobs which place frequent but small demands, can reduce the overall system average response time.[1] Such size-aware discriminatory scheduling algorithms have been shown, both experimentally and analytically (see[2] and references therein), to work extremely well when the job size distribution possesses the heavy-tailed (HT) property[†]. Since data transfer in a network can be modeled as a flow scheduling problem, and the HT property has been observed in the length of Internet transactions, especially Web file transfers, it is natural to design a network system that favors short file transfers.

As a result, before starting transmission of a flow[‡], the network has to know the length of each flow in advance. Such information may not be readily available (e.g. as for dynamic web pages). Even if flow lengths are available, it may not be desirable to propagate this information to the network to keep it independent of application semantics.[3] Instead, one can let the network implicitly identify short flows by "testing" their status. Specifically, we assume some traffic controller inside the network that measures how much traffic a flow had inserted into the network. Henceforth, we refer to such measure as the "age" of a flow. Young (new) flows are always assigned to the highest priority. Once a flow's age exceeds a certain threshold, its priority is reduced and the data transfer rate becomes slower than that allocated to other "younger" flows.

In this paper, we model flow transfer in the Internet as a time-shared system. We apply a conservation law by Kleinrock[1] to solve for the average response time in such system. Our approach takes into account TCP congestion control, under which the average flow transfer time takes a very complicated form. Resorting to the conservation law, we are able to numerically solve for the response function under different load conditions. Our results are shown to be accurate compared to those obtained from the *ns*-2 simulator, which runs many orders of magnitude slower.

In the next section, we introduce some recent work related to the design and analysis of such discriminatory system. In Section 3 we introduce a prototype system of the proposed differentiated control scheme and present a numerical analysis which takes into account the TCP congestion control algorithm. The accuracy of the analytical method is

*This work was supported in part by NSF grant ANI-0095988.

†Note that the HT property only requires that the largest small number (say 10%) of jobs contribute most (say 90%) of the load to the system. This is different from the classical definition of heavy-tailed distribution, which requires the tail of the distribution to be of the power-law form.

‡A flow is generally defined as a sequence of packets that share certain common properties. Here a flow refers to data packets belonging to the same transaction.

shown by comparison against results obtained using the *ns*-2 simulator[4] in Section 4. We show how such analytical model may be used for traffic engineering purposes in Section 5. We conclude our paper in Section 6 with possible ways of extending our analytical model to more complicated systems.

2. RELATED WORK

The advantage of giving high priority to short jobs has been studied thoroughly in the past decades. Most of the research has focused on optimal scheduling policies like Shortest-Remaining-Processing-Time first (SRPT) or Shortest-Job-First (SJF) (see[2] and references therein). To implement these optimal policies, information such as the remaining processing time of a job at any time is needed. It may be practically impossible to maintain such information in a large-scale environment like the Internet, for which we seek a sub-optimal alternative scheduling algorithm.

Job-size aware scheduling has been implemented in end-systems. Recently,[5] Harchol-Balter *et al.* implement the SRPT scheduling algorithm in a Linux web server. They then argue that since the file distributions in most web servers possess the HT property, such implementation can significantly enhance the system performance. However, to observe such enhancement, they have to assume that the bottleneck is at the outgoing interface of the web server, which is not always true in a wide-area network environment. In other words, preferential treatment to short jobs has to be provided inside the network as well for the sake of *end-to-end* response time. Moreover, since the number of priority levels in a machine is *finite*, their implementation is only an approximation of the real SRPT algorithm. They do not analyze this approximation scheme. Yang and de Veciana[6] propose an optimal size-aware bandwidth allocation scheme which gives higher transmission rate to short TCP transactions. In their scheme, end-system TCP slows down its window increase rate once the session size exceeds a certain threshold. Therefore, without modification to TCP at *every* end-user, clients who are equipped with the new congestion control algorithm may lose bandwidth to those who are not. In their analysis, they formalize the bandwidth sharing problem as an optimization problem. The objective is to minimize the average response time for the entire system. They did not give in closed-form the response time for different sizes.

On the contrary, our objective in this paper is to compute the average response time for each individual flow size under general flow size distributions. Moreover, our numerical analysis considers the effect of TCP congestion control on flow scheduling. Therefore our results closely match those obtained from *ns*-2,[4] a packet-level simulator.

3. ANALYZING A DISCRIMINATORY SYSTEM WITH TCP FLOWS

The goal of a next generation transport layer protocol is to fairly and efficiently share network resources (bandwidth). Therefore, in the ideal case, network flow scheduling can be modeled as a processor sharing system. Unfortunately, the current dominant transport layer protocol, TCP, can only statistically achieve idealized bandwidth sharing, when the size of flows is long enough (see[7] and references therein). On the contrary, due to the conservative nature of TCP congestion control, short TCP flows usually receive less share than long flows when competing for bandwidth under the same conditions.[8] To enhance the performance of short TCP flows, we propose in[8] to reduce the loss rate seen by short flow packets.

3.1. A Flow-size Aware Discriminatory Network System for Heavy-tailed Traffic

We use an architecture similar to the Differentiated Services model to build our traffic management system. Figure 1 illustrates the architecture of the system.

In such system, the network is divided into domains. Routers at the edge of the domains (namely Internet Traffic Managers) are capable of maintaining per-flow state, so that they can classify flows according to their characteristics (e.g. long or short). Routers inside the domain (namely core routers) only differentiate incoming packets according to which class they belong to. We call such system *flow-size aware discriminatory* policy to distinguish it from traditional *flow-size oblivious* network scheduling policies. The differentiation is realized by Active Queue Management (AQM) (e.g., see examples in[9]). However, without violating end-to-end design principles,[3] edge routers do not have detailed information such as how long the flow will stay in the network. Therefore, we resort to the following heuristics to give preferential treatment to short flows. By default, packets from a newly arriving flow are given the highest priority. Each edge router maintains a counter for every active flow to account for how many packets (or bytes) have been

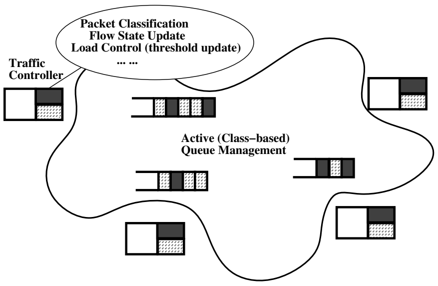


Figure 1. Proposed Architecture

received so far[§]. Once the counter exceeds a certain predefined threshold, the remaining packets from the same flow will be marked as belonging to the next lower priority level. The core routers can then treat packets of different classes accordingly to achieve service differentiation.

For ease of analysis, we assume that there are only two classes of flows. Let p denote the drop (or mark in case ECN is used[10]) rate at which core routers drop incoming high priority packets and denote by q the drop (mark) rate for low priority packets. From the TCP friendly equation,[11] when loss rate is small, TCP throughput is inversely proportional to the square root of loss rate. Thus, if $q = wp$, and $q \ll 1$, the throughput of a high priority flow is roughly $\sqrt{w}$ times that of a low priority flow. If we let $w \to \infty$ (or equivalently $p \to 0$), then the bottleneck queue effectively becomes a priority queue, i.e., low priority packets can not be forwarded as long as there are backlogs of high priority packets. In Diffserv terminology, the former model is called the Proportional Diffserv[12] while the latter one (with $w \to \infty$) is called the Expedited Services.[13]

3.2. Applying Conservation Law to Analyze TCP Performance

We use the same approach as in[14] to derive the response time of a typical TCP flow as a function of its length d, the average dropping (marking) rate p or q, and other relatively static parameters such as average round trip time RTT, average retransmission timer RTO, default initial retransmission timer ITO, and the receiver buffer size W_{max}. The analysis is outlined in Appendix A. Given network setup and static measurements, and assume q is always proportional to p, the response function is simply controlled by a single free parameter p. We may thus obtain the closed-form response time function by Kleinrock's Conservation Law, which states the following:

THEOREM 3.1. **Kleinrock's Conservation Law for Time-Shared Systems**. *For any M/G/1 system and any work-conserving queueing discipline, the average response time $T(x)$ for jobs of length x, satisfies the following equation:*

$$\int_{0^-}^{\infty} T(x)[1 - B(x)]dx \quad = \quad \frac{\overline{X^2}}{2(1-\rho)} \tag{1}$$

where ρ, $B(x)$ and $\overline{X^2}$ represent the average load of the system, the cumulative distribution of job sizes and the second moment of the job size distribution, respectively.

In summary, the Conservation Law states that, no matter how the jobs are scheduled, the average unfinished work in the system, which is written as a function of the average response time for jobs of different sizes, must be invariant.

Therefore, since $T(x)$ has only one free variable (loss rate p), we can utilize the Conservation Law to solve for the closed-form of $T(x)$, given the job size distribution and average system load.

However, the response function itself is of a very complicated form and we can not use Kleinrock's Conservation Law directly to compute it, since it is not easy to perform the integrations in Equation (1).

Nevertheless, if we assume the response time function is continuous and smooth, we can approximate the integration by its corresponding Riemann sum over discrete intervals, i.e.:

$$\int_{0^-}^{\infty} T(x)[1 - B(x)]dx \approx \sum_{i=0}^{\infty} T(\overline{Y_i}) \int_{a_i}^{a_{i+1}} [1 - B(x)]dx$$

[§]For simplicity, we assume packets are of the same size and express a flow size in terms of packets.

where $[a_i, a_{i+1}]$'s are some predefined intervals, and $\overline{Y_i}$ is the average length of flows whose length is between a_i and a_{i+1}, given by $\overline{Y_i} = \int_{a_i}^{a_{i+1}} \frac{x dB(x)}{B(a_{i+1}) - B(a_i)}$. The smaller the intervals are, the more accurate the approximation is.

We can now apply Kleinrock's Conservation Law to compute the average loss rate(s) for TCP under different network scheduling policies. We can rewrite Kleinrock's Conservation Law in the form of a root finding problem:

$$\sum_{i=0}^{\infty} T(\overline{Y_i}, p) \int_{a_i}^{a_{i+1}} [1 - B(x)]dx - \frac{\overline{X^2}}{2(1-\rho)} = 0 \tag{2}$$

Since $T(x)$ is an increasing function of the loss rate p, given the properties of the input, e.g., arrival and flow length distributions, and the values of other static parameters, there should exist a single solution for p. For the analysis of our proposed discriminatory system, although the loss rate changes depending on how many packets a flow has transferred, since we assume the change is proportional and we know the increase factor w, the root finding problem still has only one free variable (p or q).

We resort to the Bisection method to solve for the loss rate. Denoting by $LHS(p)$ the left-hand side in Equation (2), Figure 2 illustrates the process of our iterative numerical approach. We choose the values of a_i's in step 2 so that the interval grows exponentially, i.e., $a_{i+1} = (1+\epsilon)a_i$. This is because the response function $T(x)$ of TCP shows irregular form for small x but approaches a linear function as x becomes larger.

```
RootFind()
1   p_a ← 0, p_b ← 0.99, p ← (p_a + p_b)/2
2      do while |LHS(p)| > ε
3         if LHS(p) · LHS(p_a) < 0
4            then   p_b ← p
5            else   /* LHS(p) · LHS(p_b) < 0 */
6                   p_a ← p
7         p ← (p_a + p_b)/2
8     return p
```

Figure 2. Bisection method to compute loss rate

4. *NS*-2 SIMULATION RESULTS

We also implemented our proposed TCP discrimination scheme in the *ns*-2 simulator.[4] We assume network traffic is dominated by Web-like transactions. To this end, we adopt the Web traffic model of Feldmann *et al.*[15] In this model, randomly selected clients initiate sessions which involve surfing several web pages of different sizes from randomly chosen websites. Each page may contain several objects, each of which requires a TCP connection for delivery (in other words, an HTTP 1.0 model is assumed). To request a page, the client sends a request packet to the server (simulating the GET message in a typical HTTP session). The server responds with an acknowledgment and then starts to transmit the web page requested by the client. The distributions of inter-page and inter-object time (in seconds), page size and object size (in packets) are given in Table 1. The topology used in our simulation is shown in Figure 3.

Node 0 acts as the entry edge router of the bottleneck domain whereas node 1 and node 2 are core routers. All routers are assumed to be ECN (Explicit Congestion Notification) capable. That is, packets are marked rather than dropped as the router queue grows. TCP then responds to these ECN markings by adjusting its packet sending rate. The link between node 1 and node 2 is the bottleneck. Other links are configured to be lossless. The buffer size of the bottleneck link and the configuration of its queue management policy are carefully tuned according to[16] so as to maximize power[¶]. For our flow-size aware scheme, we tune the marking functions used at the core router

[¶]Power is defined as the ratio between throughput and latency, so a high power implies low delay and high throughput.

# of sessions	inter-page	objs/page	inter-obj	obj size
150	Exponential mean 8.0	Uniform min 3 max 8	Exponential mean 0.05	Bounded Pareto [4,200000] shape 1.2

Table 1. Web traffic configuration

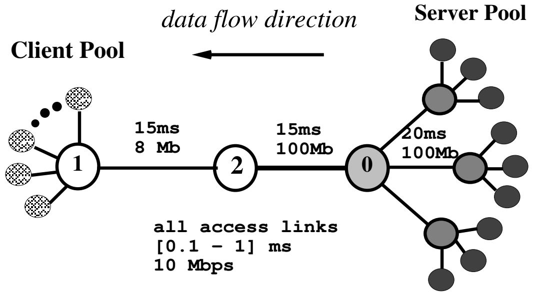


Figure 3. Simulation Topology

(router 2), so that when congestion is about to happen, packets from short flows are marked at a rate $1/w$ times that of long flows. We assume data packets are 500 bytes long and the receiver has unlimited buffer size. We choose the flow size cutoff threshold to be 50 packets. According to the definitions in Appendix A, the simulation configuration corresponds to: flow arrival rate $\lambda = 93.75$, bottleneck link capacity $C = 2000$ packet/second, average $RTT \approx 160\ ms$, $RTO \approx 4RTT$, $W_{max} = C \cdot RTT = 320\ packets$, $thr = 50$. We also choose $ITO = 1\ second$ as the initial retransmission timer in our simulation.

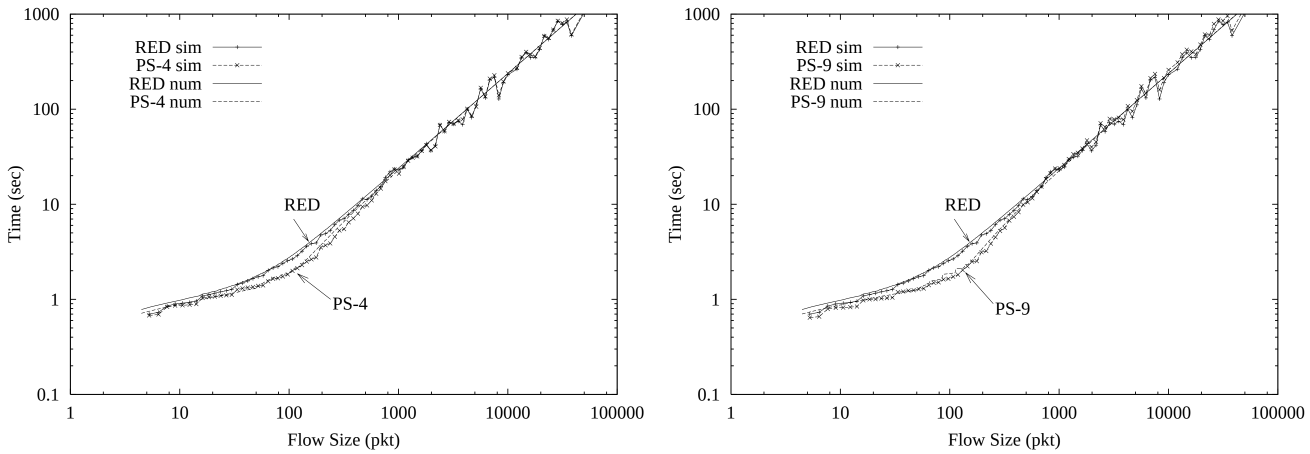
In the remainder of this section, we use the name "RED" to describe the size-oblivious system while use the term "PS-w" ($w > 1$) to refer to our size-aware scheme which gives preferential treatment to short flows. w is the weight parameter used to compute different marking probabilities. According to our numerical analysis in Section 3.2, RED corresponds to PS-1, i.e., $w = 1$ or $q = p$. Figures 4(a) and 4(b) plot the average response time versus flow size for the cases when long flow packets are marked at a rate 4 and 9 times higher than short flow packets, respectively. We also plot the corresponding numerical results for each case. Notice that, however, since we are dealing with flow size distributions with high variability, we use the measured system load (0.979) instead of $\lambda\overline{X}/C \approx 0.984$ as the load factor ρ in our numerical approach (cf. Equation 2). Under this load, the measured packet marking rate is around 2.5%.

It is evident that the numerical results match the simulation results quite well. Specifically, both the numerical analysis and the simulation illustrate that with w set to 4, our proposed differentiated control scheme reduces the response time of medium sized flows (flows containing 50-200 packets) by 50-70%, without significantly penalizing large flows (less than 3% increase in response time). When the weight factor w is increased to 9, we observe a 60-80% reduction in response time for medium sized flows and less than 5% increase for large flows[‖]. Our numerical approach runs much faster for this specific case — simulation takes about 4 hours on a 500 Mhz Pentium III processor, while our numerical approach takes less than 2 seconds.

5. LOWER BOUNDS ON RESPONSE TIME AND APPLICATION TO TRAFFIC ENGINEERING

It can also be seen from Figure 4 that it takes on average longer time for short flows to transfer the same amount of data than long flows, when flow scheduling is unaware of the length of the flow. However, even though our size-aware scheme tries to protect short flows by reducing the number of packet losses experienced by them, they still can not fully utilize the available bandwidth. This is because a new TCP flow needs a learning phase, called Slow-start, to adapt to the rate at which it can safely transfer packets without causing congestion in the network. In other words,

[‖]Notice that the response time graphs are plotted in log-log scale.



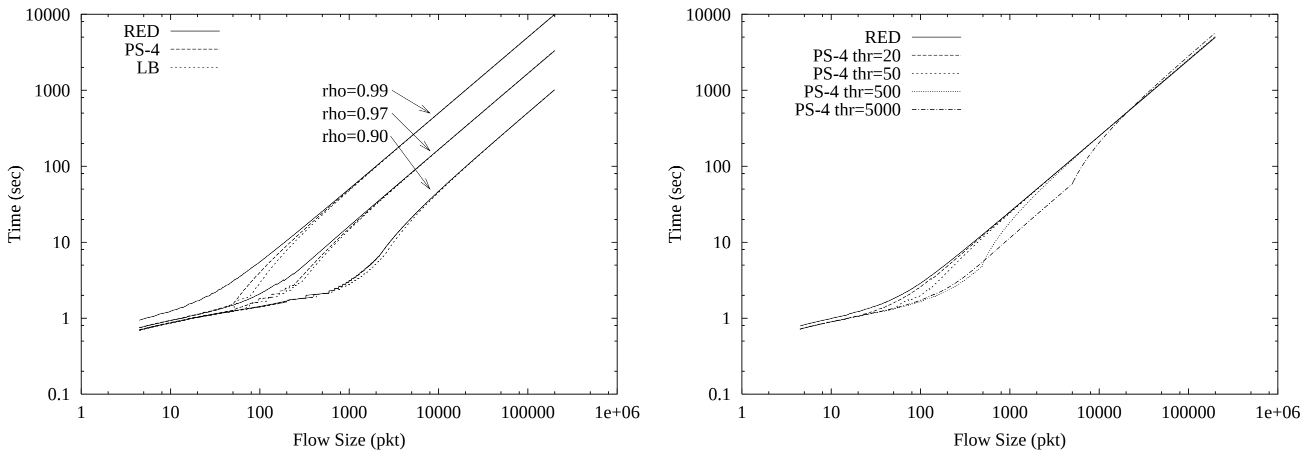
(a) $w = 4$ (b) $w = 9$

Figure 4. Average Response Time for different flow size

the average response time of a TCP flow is lower bounded by the length of such learning phase and the transient phase afterwards, as described in Appendix A.

On the other hand, since the size-oblivious RED scheme has the least discrimination, its response time can be used as the lower bound for long jobs. That is, the lower bound for a discriminatory service system is achieved when there is no drop (or ECN marking) before the cutoff threshold and the dropping (marking) rate becomes p_{RED} (the average dropping/marking rate for RED) after the threshold.

We now plot the response time function obtained by our numercial approach and the corresponding lower bounds under different load conditions and with different cutoff thresholds in Figure 5.



(a) Cutoff $thr = 50$, weight $w = 4$, varying load ρ (b) Load $\rho = 0.98$, $w = 4$, varying thr

Figure 5. Numerical Analysis of the Response Time

From Figure 5, we can make the following observations:

- Since we use a variant of the RIO management policy at the bottleneck queues, service discrimination can only be provided when there are backlogs in the queue. This condition implies that our proposed scheme is only effective *at times* when the network is highly utilized. As shown in Figure 5(a), when the average load is below 90%, the response time function of the file size-oblivious RED scheme is already very close to the lower bound.

However, with more advanced techniques such as virtual queue based management algorithms,[17] it is possible to achieve better differentiated control even at a much lower load condition.

- Increasing the cutoff threshold reduces the response time for more medium sized flows, without hurting both longer and shorter flows' performance. This is because for longer flows, the file size distribution after the cutoff still exhibits high variance, which dominates the response time function. For shorter flows, their response time is lower-bounded by the slow-start phase of TCP.
- When the load is not extremely high (e.g., less than 98%), and the cutoff threshold is not extremely high (e.g., less than 10 times the average flow size), using a weight factor of 4 (roughly 2 if measured in throughput) can almost reduce the response time for short and medium sized flows to the lower bound. As illustrated by Figure 4, both simulation and numerical results show that increasing the weight factor to 9 does not help much in reducing response times.

Our analytical model can also be used for other traffic engineering problems such as empirically finding the optimal cutoff threshold under different file size distribution and load conditions. For example, from Figure 5(b) we predict that using a cutoff threshold of 500 can reduce the response time for flows whose sizes are between 50 and 2500, without hurting much the performance of both longer and shorter flows. We reran the *ns*-2 simulation in Section 4 with cutoff threshold changed from 50 to 500 packets. The results are shown in Figure 6 and they indicate that our prediction is correct.

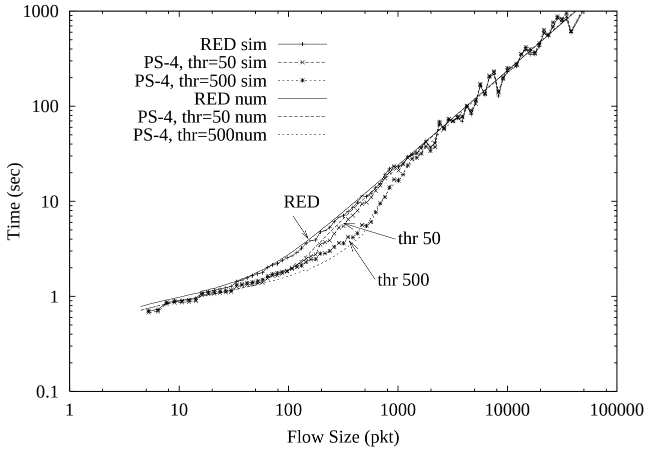


Figure 6. Simulation results for cutoff thresholds $thr = 50$ and $thr = 500$, $w = 4$,$\rho = 0.979$.

6. CONCLUSION AND FUTURE WORK

We show in this paper that Kleinrock's Conservation Law can be utilized to predict the average response time in a time-shared system like the Internet. We study the problem of how a discriminatory scheduling scheme may help enhance the average response time experienced by end-users under heavy-tailed flow sizes. We take into account the bandwidth sharing model achieved by TCP control algorithms. Our method is shown to be accurate compared to packet-level simulation, which is many orders of magnitude slower.

We are currently extending our model in the following directions:

- Packets could be lost during transmission. Since TCP provides reliable service, it tries to retransmit packets that are considered lost. The loss estimation at the sender may not be accurate. In this case, there may be redundant packet retransmissions. This makes the network system non-work-conserving since the server becomes effectively idle serving these redundant retransmissions rather than original packets. This is problematic since our analysis is based on Kleinrock's Conservation Law which requires the system to be work-conserving. In addition, if the total arrival rate (of original and redundant packets) exceeds the system capacity, the system becomes unstable.

In our simulations, the impact of redundant packet retransmissions was not so significant since we assume all network nodes are ECN-capable (i.e. packets are marked rather than dropped as the bottleneck queue grows). We are currently extending our analytical model to consider redundant packet retransmissions. The challenge is to model a redundant retransmission factor. We can then convert the non-work-conserving system back to work-conserving by extending the length of each job by this factor. The model is also being extended to account for finite queue sizes.

- Ben Fredj *et al.*[7] propose a user impatience and reattempts model to analyze the effect of interrupting and re-attempting TCP transactions. Their preliminary results show that the HT property helps increase network goodput (i.e. bandwidth used to transfer useful packets). We are extending our discriminatory system model to consider such user impatience factor. Since our discriminatory scheme enhances the performance of short to medium sized flows, we believe it will still help increase network goodput in the presence of impatient users.

REFERENCES

1. L. Kleinrock, *Queueing Systems, Volume II: Computer Applications*, ISBN 0-471-49111-1. John Wiley & Sons Inc., 1976.
2. N. Bansal and M. Harchol-Balter, "Analysis of SRPT Scheduling: Investigating Unfairness," in *Proceedings of ACM SIGMETRICS'01*, (Boston, MA), June 2001.
3. J. Saltzer, D. Reed, and D. Clark, "End-to-End Arguments in System Design," in *Proc. of Second International Conference on Distributed Computing Systems (ICDCS'81)*, pp. 509–512, April 1981.
4. E. A. et al., "UCB/LBNL/VINT Network Simulator - ns (version 2)." Available at http://http://www.isi.edu/nsnam/ns/.
5. M. Harchol-Balter, N. Bansal, B. Schroeder, and M. Agrawal, "Implementation of SRPT Scheduling in Web Servers," Tech. Rep. CMU-CS-00-170, School of Computer Science, Carnegie Mellon Univ., Pittsburgh, PA, 2001. Submitted for Publication.
6. S.-C. Yang and G. de Veciana, "Size-based Adaptive Bandwidth Allocation: Optimizing the Average QoS for Elastic Flows," in *Proceedings of IEEE INFOCOM'02*, (New York, NY), June 2002.
7. S. B. Fredj, T. Bonald, A. Proutiere, G. Régnié, and J. Roberts, "Statistical Bandwidth Sharing: A Study of Congestion at Flow Level," in *Proceedings of ACM SIGCOMM'01*, pp. 111–122, (San Diego, CA), September 2001.
8. L. Guo and I. Matta, "The War Between Mice and Elephants," in *Proc. ICNP 2001*, (Riverside, CA), November 2001.
9. S. Floyd, "Variants of RED queue." http://www.aciri.org/floyd/red.html.
10. S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review* **24(5)**, October 1994.
11. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," in *Proceedings of ACM SIGCOMM'98*, (Vancouver, Canada), September 1998.
12. C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional Differentiation and Packet Scheduling," in *Proceedings of ACM SIGCOMM'99*, (Cambridge, MA), September 1999.
13. Y. Bernet and et al, "A Framework for Differentiated Services," Internet Draft ⟨draft-ietf-diffserv-framework-02.txt⟩, Feburary 1999.
14. N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP Latency," in *Proceedings of IEEE INFOCOM'00*, (Tel Aviv, Israel), March 2000.
15. A. Feldmann, A. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP Traffic: A Study of the Role of Variability and the Impact of Control," in *Proceedings of ACM SIGCOMM'99*, (Cambridge, MA), September 1999.
16. M. Christiansen, K. Jeffay, D. Ott, and F. Smith, "Tuning RED for Web Traffic," in *Proceedings of ACM SIGCOMM'00*, (Stockholm, Sweden), August 2000.
17. S. Kunniyur and R. Srikant, "Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," in *Proceedings of ACM SIGCOMM'01*, (San Diego, CA), September 2001.
18. W. Stevens, *TCP/IP Illustrated, Vol. 1: The Protocols*, Addison-Wesley, 1997.

APPENDIX A. MODELING WEB TRANSACTION LATENCY

We adopt the same approach as in[14] to model a typical Web transaction latency. Figure 7 illustrates the process of a typical web transaction and how the TCP congestion window evolves during this process. In our derivation, we assume a Reno-type congestion control algorithm; we refer the reader to[18] for terminologies and details about TCP and Reno-style congestion control. In a Web session, the client initiates a TCP connection to the server to send a request packet. The server then responds with another (or many) TCP connection(s) to transmit a webpage. For simplicity, we assume only one TCP connection is used. Thus, a typical Web transaction latency includes the following components: the time to send a page request, the time for the server to send back the first packet (with no sampled RTT's), the time to send the rest of the packets (for simplicity, assume accurate RTT estimation).

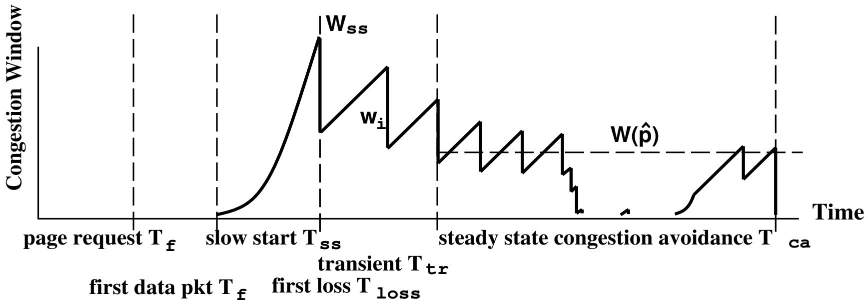


Figure 7. Web transfer over TCP illustrated

Given the session length d (in packets), average loss probability p, average path RTT (including queueing delay), default initial retransmission timer ITO, estimated retransmission timer RTO, the link capacity C, and the receiver window size W_{max}, we can derive the value of each component through a similar approach as.[14] The difference between our analysis and[14] is that we now need to take into account the change in average loss rate values due to the AQM at the core routers (i.e., switching from p to q after a certain amount of packets are transferred). Moreover, our analysis takes into account the transient stage during which the TCP congestion window migrates from a transient value to the steady-state value (T_{tr} in Figure 7).

We thus represent the transfer latency of a typical Web transaction as a function of its size and the loss rate p and the solution of Equation (2) can be obtained numerically, as shown in Section 3.2.

A.1. Page Request and First Packet of the Webpage

Loss of either the data or acknowledgment in transmitting the first packet of a TCP session requires an exponential backoff process to recover. The retransmission timer is set to the default value ITO, and for each packet loss, the timer is doubled until it reaches $64 \cdot ITO$ or a predefined upper bound (typically 120 seconds). Therefore, denote by F the random variable for the number of retransmissions, the latency to transmit the page request is the same as that to transmit the first packet of the requested webpage, and can be computed as follows:

$$E[T_f] = P[F=0] \cdot RTT + \sum_{i=1}^{6} P[F=i] \cdot \{RTT + \sum_{k=0}^{i-1} 2^k ITO\}$$

Under uniform loss, R follows a geometric distribution so $P[F=i] = p^i(1-p)$. When p is small, the above equation can be approximated by:

$$E[T_f] \approx RTT + ITO \cdot (\frac{1-p}{1-2p} - 1) \tag{3}$$

A.2. Time to Transfer Remaining Packets

Transmitting the remaining packets may span the following stages: slow-start, first packet loss, transient stage between slow-start and steady-state, steady-state (congestion avoidance). Recall that in our proposed scheme, a long TCP session may experience different loss rate depending on how many packets it has transferred. For simplicity, we illustrate the derivation of the transmission time of a two-level system here. That is, once a TCP connection transmits more than thr packets, the loss rate changes from p to q. Similar to,[14] the expected number of packets to send in slow-start, denoted by $E[d_{ss}]$, is given by:

$$\begin{aligned} E[d_{ss}] &= (\sum_{k=0}^{max(thr-1,d-1)} (1-p)^k \cdot p \cdot k) + (\sum_{k=thr}^{d-1} (1-p)^{thr-1} \cdot (1-q)^{k-thr} \cdot q \cdot k) \\ &\quad + (1-p)^{max(thr,d)} \cdot (1-q)^{max(0,d-thr)} \cdot d \\ &= \begin{cases} \frac{(1-(1-p)^d)(1-p)}{p} & d \le thr \\ \frac{(1-(1-p)^{thr})(1-p)}{p} - thr(1-p)^{thr} + \frac{(1-p)^{thr}(thr+(1-thr)(1-q)-(1-q)^{d-thr+1})}{q} & d > thr \end{cases} \end{aligned} \tag{4}$$

The separated terms correspond to whether the first packet loss occurs before or after thr has been transferred, thus different loss rate (p or q) will be used.

Thus the expected window size when slow-start finishes is given by:

$$E[W_{ss}] = \frac{E[d_{ss}]+1}{2} \tag{5}$$

And the time TCP spends in the slow-start stage is[14]:

$$E[T_{ss}] = \begin{cases} RTT \cdot [\log_2(W_{max}) + 1 + \frac{1}{W_{max}}(E[d_{ss}] - 2W_{max} + 1)] & \text{when } E[W_{ss}] > W_{max} \\ RTT \cdot \log_2(E[d_{ss}]+1) & \text{when } E[W_{ss}] \le W_{max} \end{cases} \tag{6}$$

The end of slow-start is marked by the first packet loss, and the time to recover the first packet loss is given by[14]:

$$T_{loss} = l_{ss} \cdot (Q(\hat{p}, E[W_{ss}]) \cdot E[Z^{TO}] + (1 - Q(\hat{p}, E[W_{ss}])) \cdot RTT) \tag{7}$$

where l_{ss} is the probability of having a packet loss, $Q(.)$ denotes the conditional probability that such loss is detected by timeout, and Z^{TO} is the time required to recover a timeout loss. They are given by:

$$l_{ss} = \begin{cases} 1-(1-p)^d & d \le thr \\ 1-(1-p)^{thr}(1-q)^{d-thr} & d > thr \end{cases} \tag{8}$$

$$Q(p,w) = min(1, \frac{(1+(1-p)^3(1-(1-p)^{w-3}))}{(1-(1-p)^w)/(1-(1-p)^3)}) \approx min(1, \frac{3}{w}) \tag{9}$$

$$E(Z^{TO}) = G(p) \cdot RTO \tag{10}$$

$$G(p) = (1-p)[\sum_{k=1}^{6} p^{k-1}(2^k-1) + \sum_{k=7}^{\infty} p^{k-1}(63+64(k-6))] \approx \frac{1}{1-2p} \tag{11}$$

and $\hat{p}$ is the loss rate value (p or q) which depends on whether TCP has delivered more than thr packets.

When slow-start finishes, the TCP congestion control algorithm eventually brings the congestion window to the steady-state value. That is, by the well-known TCP-friendly equation.[11,14] That is, the expected congestion window size, $W(p)$, and the steady-state throughput, R, are given by:

$$W(p) = 1 + \sqrt{(\frac{8(1-p)}{3p}+1)} \tag{12}$$

$$R = \begin{cases} \dfrac{\frac{1-\hat{p}}{\hat{p}} + \frac{W(\hat{p})}{2} + Q(\hat{p}, W(\hat{p}))}{RTT(\frac{W(\hat{p})}{2}+1) + RTO\frac{Q(\hat{p},W(\hat{p}))G(\hat{p})}{1-\hat{p}}} & \text{if } W(\hat{p}) < W_{max} \\ \dfrac{\frac{1-\hat{p}}{\hat{p}} + \frac{W_{max}}{2} + Q(\hat{p}, W_{max})}{RTT(\frac{W_{max}}{8} + \frac{1-\hat{p}}{\hat{p}W_{max}} + 2) + RTO\frac{Q(\hat{p},W_{max})G(\hat{p})}{1-\hat{p}}} & \text{otherwise} \end{cases} \tag{13}$$

Notice that the value of steady-state average window size $W(p)$ is different from that immediately after slow-start W_{ss}. Sometimes the difference is large, especially under our proposed scheme in which loss rate in the slow-start phase is much smaller than that afterwards. Therefore, we need to take into account the data transfer pattern during such transient stage. During this stage, TCP relies on the additive increase, multiplicative decrease (AIMD) algorithm to stabilize its window size. Assume after $i \cdot RTT$, the TCP window size value is w_i, then the expected window size in the next RTT is:

$$\begin{aligned} w_1 &= \frac{E[W_{ss}]}{2} \\ E[w_{i+1}] &= (1 - P[\text{no loss}])(Q(\hat{p}, w_i) \cdot 1 + (1 - Q(\hat{p}, w_i)) \cdot \frac{w_i}{2}) + P[\text{no loss}](w_i + 1) \\ P[\text{no loss}] &= (1-\hat{p})^{w_i} \end{aligned}$$

that is, the size of congestion window in the next round depends on whether a packet from previous window is lost and how the loss is detected.

The number of packets delivered after k stages is:

$$E[d_k] = \sum_{i=1}^{k} E[w_i]$$

Let m be the number of rounds it takes to converge to the steady-state window size $W(\hat{p})$, or to finish the total data transfer, i.e.:

$$m = \min_k \{E[w_k] \geq W(\hat{p}), E[d_k] > (d - E[d_{ss}])\}$$

The number of packets delivered after this stage is:

$$E[d_{tr}] = \min(\sum_{i=1}^{m} E[w_i], d - E[d_{ss}])$$

Therefore, $E[T_{tr}]$, the time spent in this transient stage is:

$$E[T_{tr}] \approx m \cdot RTT \tag{14}$$

And the time required to finish transmitting the remaining packets is:

$$E[T_{ca}] = \frac{\min(0, d - E[d_{ss}] - E[d_{tr}])}{R} \tag{15}$$

We can now write down the expected time for a client to download a webpage of size d packets as the combination of the above equations as:

$$E[T] = E[T_f] + E[T_f] + E[T_{ss}] + E[T_{loss}] + E[T_{tr}] + E[T_{ca}] \tag{16}$$