Challenges to Congestion Control Posed by Concurrent Downloads

Yong Liu [1] Weibo Gong [2]

Abstract

Concurrent downloads open multiple parallel connections to improve users' download latency. They break at user level the fairness which congestion control schemes try to maintain at connection level. They also change the network traffic statistics and challenge the performance of congestion control schemes. In this paper, we study concurrent downloads' challenges to congestion control system's fairness and transient behavior.

1 Introduction

In the past decade, communication networks have experienced dramatic growth in all dimensions: size, speed, heterogeneity, applications and users, etc. In a stochastically shared network, such as the Internet, congestion is inevitable and is the key factor determining the quality of service perceived by end users. In order to avoid congestion collapse, users must be responsive to congestion within the network. Different end-to-end congestion control schemes, such as TCP, have been implemented to assure the stability of networks. In those schemes, users adapt their transmission rates based on congestion indications along their routes. Recent studies have shown that pure end-to-end congestion control is not sufficient to achieve high network utilization and good performance for end users. Active Queue Management (AQM) algorithms, e.g. PI Controller [11], AVQ [12], RED [9], REM [1], have been proposed to communicate the congestion information to end systems. The congestion information can be used by their rate adaption schemes to cooperatively drive the network to a good operating point.

From control system point of view, it is crucial for the designers of congestion controllers to understand well dynamics of control plants. Due to the complexity of networks, network congestion is different from any traditional control plant. Congestion at one node is normally caused by traffic of multiple heterogeneous users. Fast evolutions in both network architectures and users' traffic patterns make congestion within networks change rapidly. In this paper, we investigate some challenges to congestion control posed by an emerging trend in network applications, concurrent downloading. This discussion is not meant to argue against applying control theory to network congestion control. Instead, the purpose is really to share our concerns about developments of new network applications which should be taken into considerations in the design of effective congestion control schemes.

We first briefly describe several network applications using concurrent downloading in Section 2. We then present some of our results in [13] regarding the fairness issue brought up by concurrent downloads. In Section 4, we investigate how concurrent downloads challenge the performance of end-host congestion control schemes and AQM algorithms. Simulation results are presented in Section 5. We discuss concurrent downloads' potential threats to the Internet and some possible counter measures in Section 6. Section 7 concludes this paper.

2 Concurrent Downloading

Concurrent downloading means using multiple connections to download objects concurrently. There are three forms of concurrent downloading commonly employed by network applications. The first one is concurrent downloading of HTML in-line objects. To do so, the browser first downloads the requested HTML page from the server and then opens several simultaneous HTTP connections to download all remaining objects (e.g., images) embedded within the web page. Such parallel HTTP downloads improve the overall latency of accessing the web page and its constituent images. Because normal HTML in-line objects are small, its contribution to network congestion is not dramatic. With new HTTP 1.1 protocol, all in-line objects are downloaded sequentially by one persistent TCP connection. We are not concerned much by this form of concurrent downloading.

The second form of concurrent downloading is parallel downloading of segments of one object from one site. HTTP protocol allows a byte range to be specified with each request. Some applications (e.g., FlashGet [16]) have been developed to parallelize the download of each web object by opening multiple connections per object and downloading a different portion of the object on each connection. By doing so, they claim to greatly speedup HTTP downloads. The degree of download concurrency is application dependent. Currently FlashGet allows 10 connections for one object.

The third form is commonly used by emerging Peer-to-Peer applications. Basically a user of Peer-to-Peer network sends out a query for a wanted object. A list of peers who have the object will be sent back to him. The user then cuts the object into segments and sets up one connection with each peer on the list to download one segment. The Cut-

[1]Computer Science Department, University of Massachusetts, Amherst, MA 01003, U.S.A. e-mail: yongliu@cs.umass.edu

[2]ECE Department, University of Massachusetts, Amherst, MA 01003, U.S.A. e-mail: gong@ecs.umass.edu

Download process is done dynamically. During the downloading, if the user find more peers to download from, it will cut the object finer and launch more connections to download. The degree of download concurrency is dependent on the availability of the requested object on the network. For some popular objects, it is easy to find more than 10 peers to download from.

Recent Internet traffic studies show HTTP and Peer-to-Peer traffic dominate network traffic. Concurrent downloads, especially the second and the third form, have the potential to dramatically change network traffic patterns.

3 Fairness Issue

Network resources are shared among heterogeneous users. There is no explicit information about how much bandwidth is available for each individual user. As an end-to-end congestion control mechanism, TCP aims at probing and grabbing available network bandwidth and remains responsive to network congestion at the same time. It increases its sending rate additively when there is no congestion and decreases its rate multiplicatively upon receiving congestion indication from the network. It is proved in [4] that the Additive Increase and Multiplicative Decrease (AIMD) algorithm can drive the network to a fair state. In a homogeneous network, each competing TCP connection will get an equal share of bandwidth.

By employing multiple connections for one download, concurrent downloads break at application level the fairness that TCP tries to maintain at connection level. In a homogeneous setting, a user's bandwidth share is proportional to the number of connections he employed. For a more general network setting, when all connections launched by a user follow the same route, the fairness among users is formulated in [13] as a constrained optimization problem. Let $\mathcal{L}$ be the set of links in the network. The capacity of link l is c_l, $l \in \mathcal{L}$. Let $\mathcal{U}$ be the set of users, with size $|\mathcal{U}| = m$. Suppose user j launches n_j AIMD concurrent connections with additive increase step size α_j, multiplicative decrease factor β_j and round trip time τ_j. Define users' routing matrix $B = (B_{l,j}, l \in \mathcal{L}, j \in \mathcal{U})$, such that $B_{l,j} = 1$, if user j traverses link l, and $B_{l,j} = 0$, otherwise. Also assume users receive network negative feedbacks, e.g., losses, marks, at a rate proportional to its sending rate. Then in the steady state users' rates $y = \{y_j, j \in \mathcal{U}\}$ are distributed according to the solution of the following constrained optimization problem

$$\max_y F_A^u(y) = \sum_{j \in \mathcal{U}} \frac{n_j}{\tau_j} \log \frac{y_j}{n_j \alpha_j + \beta_j y_j} \tag{1}$$

subject to the constraints

$$\sum_{j \in \mathcal{U}} B_{l,j} y_j \le c_l, \ \forall l \in \mathcal{L} \tag{2}$$

From equation (1), user's degree of download concurrency n_j plays an important in determining his throughput in the network. The more connections a user employs, the high throughput he achieves.

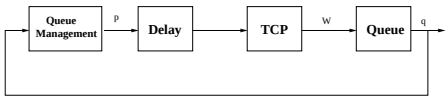


Figure 1: Control Loop of Network Congestion Control

4 Impact on the Performance of Congestion Control

Besides fairness in steady state, concurrent downloads also affect the transient behavior and performance of existing congestion control mechanisms. In this section, we systematically analyze concurrent downloads' impact on congestion control schemes

4.1 Model of Congestion Control System

Recently, it has become a very active research area to model network congestion control as closed loop control system and use methodologies in system and control area to develop better congestion control algorithms [10, 11, 1, 12]. For a single bottle-neck network, control block diagram of its congestion control system can be depicted as in Figure 1. The queue management scheme can be viewed as the controller of the closed loop control system. It measures the backlog q at the congested buffer and calculates the packet dropping or marking probability p. Packets are dropped or marked with probability p to indicate congestion to end-hosts. When congestion indications reach end-hosts after certain network delay, TCP will regulate its sending rate by adjust its congestion window w, which eventually will affect the backlog q at the congested node.

To study the performance of congestion control loop, we need to model the dynamics of TCP, congested queues and AQM schemes. Internet traffic studies observed elephants-mice classification of connections, namely a small number of long lived connections, or elephant connections, account for the majority of the total traffic volume while a large number of short lived connections, or so called mice connections, only contribute a small portion of traffic [5, 3, 6]. By assuming this, we can focus on elephant connections and treat those mice connections as noise to the control system. In [14], a non-linear model of a network of AQM routers supporting TCP flows has been established. When the loss probability within the network is small, a TCP connection works in congestion avoidance stage. Let $W(t)$ be its expected congestion window size, $R(t)$ be its the round trip time, which consists of queueing delay and propagation delay, $p(t)$ be the packet drop probability. Then the non-linear differential equation describing the evolution of $W(t)$ is:

$$\frac{dW(t)}{dt} = \frac{1}{R(t)} - \frac{W(t)W(t - R(t))}{2R(t - R(t))} p(t - R(t)) \tag{3}$$

For a single bottle-neck network, the model for the congested queue is simply

$$\frac{dq(t)}{dt} = \frac{W(t)}{R(t)} N(t) - C \tag{4}$$

where $N(t)$ is the number of connections at time t and C is the capacity of the bottle-neck link. When the system is stable, it works around its operating point $\{W_0, p_0, R_0\}$,

where W_0 is stationary TCP window size and p_0, R_0 is stationary loss probability and round trip time. We have

$$W_0 = \frac{R_0 C}{N} \tag{5}$$

$$p_0 = \frac{2N^2}{R_0^2 C^2} \tag{6}$$

At steady state, average TCP window size is inversely proportional to the number of connections N and packet loss probability is proportional to N^2.

By carrying out linearization [10] around the system's operating point, we can obtain the linear model of the congestion control system. The transfer function of TCP is

$$P_{tcp}(s) = \frac{\frac{R_0 C^2}{2N^2}}{s + \frac{2N}{R_0^2 C}} \tag{7}$$

The transfer function of the bottle-neck queue is

$$P_{queue}(s) = \frac{\frac{N}{R_0}}{s + \frac{1}{R_0}} \tag{8}$$

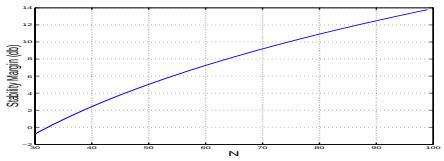
The overall transfer function of the control plant is

$$P(s) = P_{tcp}(s) P_{queue}(s) P_{delay}(s) = \frac{\frac{C^2}{2N} e^{-sR_0}}{(s + \frac{2N}{R_0^2 C})(s + \frac{1}{R_0})} \tag{9}$$

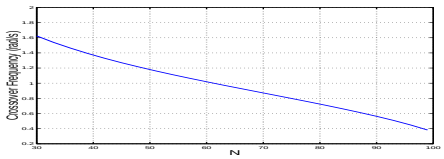
4.2 Transient Behavior

Concurrent downloads increase the number of concurrent connections networks. If every user employs m connections for his download, the number of connections in a network will increase by a factor of m. From (9), gain of the congestion control loop is inversely proportional the number of connections in system. One counter intuitive consequence is that the more connections in the system the more stable the system (See remark 3(5) in [10]). This is because at steady state congestion window size of each connection is inversely proportional to N. Impact of N on TCP window size's sensitivity toward packet loss probability p is two fold. First, the loss event arrival rate at each connection is proportional to its current window size; Secondly, TCP window back off after receiving a loss is proportional to its window size. On the other hand, the sensitivity of queue length toward expected TCP window size is only proportional to N. The overall sensitivity of the control plant $P(s)$ is inversely proportional to N. Figure 2 shows that a congestion control loop's stability margin increases while its $0db$ crossover frequency decreases when the number of connections N increases. With more TCP connections in the system, the congestion control loop tends to be more stable and yet more sluggish. Experiments in Section 5 will show concurrent downloads change control loop's responsiveness and stability significantly.

If there are too many concurrent connections, according to (6), network will operate in high loss rate state. It has been pointed out in [15] that TCP is not scalable with the number of concurrent TCP connections. As a window based congestion control scheme, TCP has no mechanism to send out less than one packet per round trip time other than



(a) Stability Margin



(b) Crossover Frequency

Figure 2: System Stability and Responsiveness against the Number of Concurrent Connections

time-out. In time-out stage, TCP behaviors are quite different from in congestion avoidance stage. Nonlinear dynamic model in (3) can no longer accurately capture TCP window dynamics. It is difficult for AQM schemes designed out of the linearized model to control the queue. In addition, frequent time-outs will also greatly reduce TCP's throughput. Packets lost in the network have to be retransmitted. Thus the good put of each connection decreases.

4.3 Network Load Level Variation

Concurrent downloads not only increase the average number of current connections in a network but also make network load level, in terms of number of active connections, change more dramatically and more frequently. For sophisticated AQM schemes, such as PI Controller [11] and RED [9], parameter tuning is crucial for their performance. A very important factor to consider when tuning their parameters is the network load level. Some new AQM schemes have been proposed to adaptively adjust their parameters in response to load level change [17, 8, 12]. To achieve good performance, it is important for AQM schemes to have enough time to converge to their desired steady states.

Assuming elephants-mice classification of network connections, closed loop congestion control model in Section 4.1 only focuses on elephant connections. By definition, the number of elephant connections changes slowly and mildly. In that case, it is possible for AQM controllers to adapt to network load level change. This assumption is challenged by concurrent downloads, which chop each of those elephant connections into multiple parallel shorter connections. For example, Flashget enables one to open 10 concurrent connections for one object. This will change the size distribution of network connections and challenge elephants-mice classification of network connections.

To illustrate, assume when each user employs only one connection for his download, the Complementary Cumulative

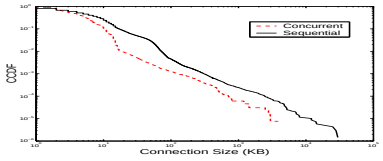
Distribution Function (C.C.D.F) of connection sizes is

$$\bar{F}_X(x) \triangleq P(\text{Connection Size is bigger than } x)$$

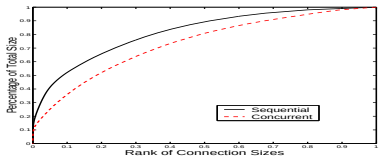
And now suppose concurrent downloads chop all connections with size bigger than K into D concurrent connections, then the reshaped C.C.D.F of connection sizes can be derived as:

$$\bar{F}_Y(y) = \begin{cases} \frac{\bar{F}_X(y)+(D-1)\times\bar{F}_X(K)}{1+(D-1)\bar{F}_X(K)} & y < \frac{K}{D} \\ \frac{\bar{F}_X(y)-\bar{F}_X(K)+D\times\bar{F}_X(D\times y)}{1+(D-1)F_X(K)} & \frac{K}{D} \le y < K \\ \frac{D\times\bar{F}_X(D\times y)}{1+(D-1)\bar{F}_X(K)} & K \le y \end{cases} \quad (10)$$

To see concurrent downloads' impact on durations of network connections, we compare connection size distribution of concurrent downloading with that of sequential downloading in Figure 3. We take an empirical file size distribution from a running cricket-info web server. The upper figure plots the connection size distribution for both sequential downloading and concurrent downloading with parameter $K = 100KB$ and $D = 10$. We can see distribution of connection size changes a lot when concurrent downloading is employed. The bottom figure shows the distribution of traffic volume across connections. With sequential downloading, about top 10% biggest connections accounts for more than half of the total traffic. With concurrent downloading, one half of the total traffic is generated by top 20% biggest connections. Concurrent downloads make traffic more evenly spread out among connections. The elephants-mice classification of web connections is less likely to be true.



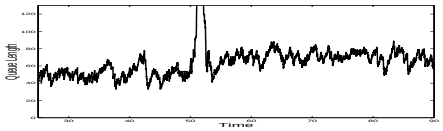
(a) Connection Size Distribution



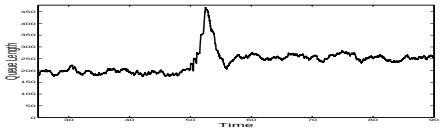
(b) Volume Distribution among Connections

Figure 3: Concurrent Downloads' Impact on Network Connection Size

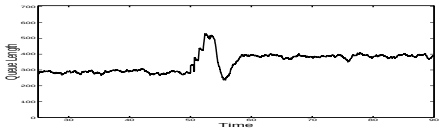
Concurrent downloads increase the number of network connections and in the same time reduce the durations of individual connections. They make the network load level change more frequently. In addition, all parallel connections of one download are closely spaced in time. Network traffic will become extremely bursty if concurrent downloads are widely employed. AQM schemes have less time



(a) $G_1 = 10$, $G_2 = 0$



(b) $G_1 = 5$, $G_2 = 5$



(c) $G_1 = 0$, $G_2 = 10$

Figure 4: Queueing Behavior

to adapt to bigger jumps in load level. This will put higher requirements on AQM's responsiveness and adaptiveness. Experiments in Section 5 will show concurrent downloads' impact on network load level variation and AQM's performance.

5 Simulations

We conducted `ns` [7] simulations to demonstrate how concurrent downloads change congestion control loop's transient behaviors and load level variation.

5.1 Experiment on Transient Behavior

The first experiment is to demonstrate concurrent downloads' impact on network's transient behavior. 10 users of a content server share a bottleneck link of bandwidth 2.4Mb/s. Each user employs either one or ten connections for their downloading. Define G_1 to be the number of users employing a single connection and G_2 to be the number of concurrent downloading users. In this experiment, we vary G_1 from 0 to 10 while keeping $G_1 + G_2$ at 10. In order to test congestion control loop's responsiveness, we increase user population by 50% at time 50 seconds.

Figure 4 compares the bottle-neck queue evolution under different user profiles. The scale of each subfigure is chosen to be proportional to the mean value of its queue length. The queue oscillation ratio gets smaller when more users employ concurrent downloading. This agrees with our stability analysis in Section 4.2. In the same time, the control loop gets more sluggish. It takes longer for the network to settle down to its operation point after load increase at time 50 seconds. Figure 5 shows the loss rate at the bottleneck link. Concurrent downloads increase packet loss rate

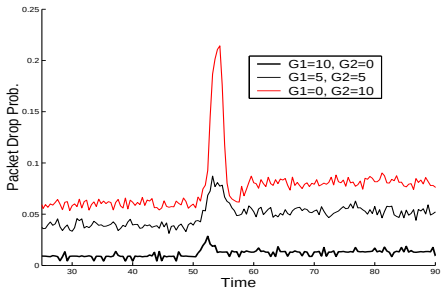


Figure 5: Impact on Packet Loss Probability

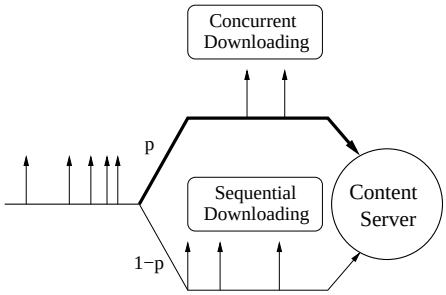


Figure 6: Content Server with Poisson Arrival

within the network. In the extreme case, when all users do concurrent downloading, the loss rate exceeds 8 percent, which is rare in a well engineered network. We observed much more TCP time-outs than when everybody uses a single connection for downloading. The link goodput and thus user download latency degrade.

5.2 Experiment on Load Level Variation

In order to show how concurrent downloading changes the load level within the network, we did another experiment with `ns`. The simulation model is depicted in Figure 6. Users arrive to a content server according to a Poisson process. Each user requests an object with size 4MB, the size of a normal MPEG-3 audio data file. The arrival rate is 0.25/sec which makes the utilization of the 10Mbps link to be 0.8. In order to compare users' download latency, we assume they all share the same round trip time of 60ms. Upon arrival, each user chooses to use 10 concurrent connections for his downloading with probability p and otherwise sticks with sequential downloading.

In this experiment, we change p from 0 to 1. For each p, we simulate the system for 8 hours and monitor the number of active connections traversing the bottleneck link. For each user, we record his start and finish time to calculate his download latency. Figure 7 plots the load level sample paths for three different concurrent downloading probabilities. Detailed statistics are presented in Table 1. Concurrent downloading not only increase average load level of the system, but also make the load level change more frequently and more drastically.

Figure 8 plots users' download latency. When p gets bigger, it takes both concurrent downloading users and single connection users longer to finish their downloads. The average download latency of all users is larger when p is bigger. Concurrent downloads degrade both TCP connections' throughput and the bottleneck link's utilization.

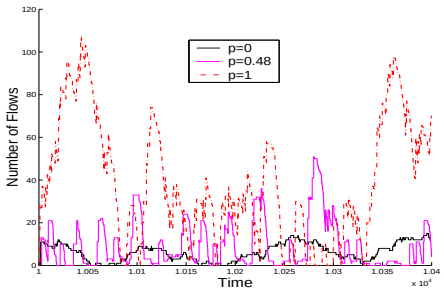


Figure 7: Evolution of Load Level: $[10000, 10400]$

Table 1: Number of Active Connections

	p=0	p=0.48	p=1.0
Average	5.93	17.5	88.63
Std. Deviation	4.69	16.2	86.77
Jumps	14326	44551	79038

6 Discussions and Open Problems

In previous sections, we have studied the concurrent downloads' fairness issue and their impact on performance of network congestion control mechanisms. From users' point of view, concurrent downloading enables them to more aggressively grab bandwidth from the network. They have incentives to employ this technique to download faster. But we have shown that too many concurrent downloads degrade the performance of the whole network. Eventually it will increase every user's download latency. Unfortunately, most users do not care about the performance of the whole network and are unaware of this situation. Even if they know too many concurrent downloads will hurt their own performance, they won't stop doing it because they don't want to be taken advantage of by other users who are doing concurrent downloading. This phenomena is called "Tragedy of Common" in game theory. Same situation is faced by competing content servers. They compete with each other for network bandwidth and users. Servers supporting concurrent downloading can attract users by claiming faster data transferring. Both end users and servers have strong

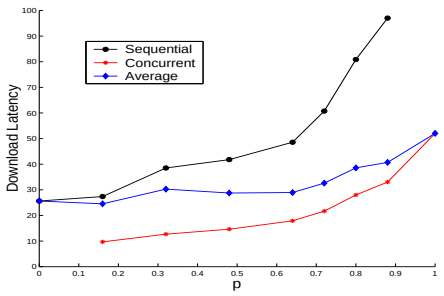


Figure 8: Users' Download Latency

incentive to use concurrent downloading. It could get more popular in network applications.

As concurrent downloads spread out, problems we discussed here will become serious. Some counter measures are needed to limit their damage to the network. One natural thinking is to do some control at the application level. Technically, web servers can disable concurrent downloading by denying connections requesting for a segment of an object. Some web servers, which have already experienced some performance problems due to too many concurrent downloads, only accept single connection sequential downloads. As we have seen before, concurrent downloading helps to improve the resource utilization and end users' download latency when the load level within the network is not high. We may not want to totally disable it. It is possible for servers to dynamically adjust the number of concurrent connections that one user can possibly employ for his downloading according to servers' workload and congestion level within the network. The problem for control at server side is that servers won't have much motivation to do it unless the number of concurrent connections goes beyond their own processing capacities. This method doesn't apply to Peer-to-Peer network, where each user only open one connection with one peer.

Another direction to address this problem is to change congestion control at the transport layer. Current TCP works on the units of connections. New congestion management architecture has been proposed to do congestion control for flow aggregates [2]. Connections within an aggregate share congestion information and regulate their sending rate cooperatively. If we put all connections initiated by one user in the same aggregate, the impact of those concurrent connections on other users can be well regulated. But it requires fundamental change of network's congestion control architecture. Moreover, how to aggregate traffic and allocate bandwidth among aggregates are problems need to be solved before it can be deployed.

One last measure we can resort to is pricing. The Internet is very limited in regulating its users' behavior. Concurrent downloading is just one simple demonstration of how easily individuals can cheat on the network. Current pricing mechanisms, e.g. flat rate charge, charge for data volume, don't punish concurrent downloading users more than single connection users. To avoid serious network congestion, users should be charged according to their contribution to the congestion of the network.

7 Conclusions

In this paper, we studied several congestion control issues brought up by concurrent downloads, an emerging trend in network applications. Fairness between users with different downloading concurrency has been formulated as an optimization problem. We studied concurrent downloads' impact on network transient behavior. Our analysis is supported by experiments. We pointed out concurrent downloads' potential threats to proper use of the Internet and provided some discussions about possible counter measures.

References

[1] Athuraliya, S., Li, V.H., Low, S.H., and Yin, Q. REM: Active queue management. In *Proceedings of the 17th International Teletraffic Congress* (September 2001).

[2] Balakrishnan, H., Rahul, H., and Seshan, S. An integrated congestion management architecture for Internet hosts. In *Proceedings of ACM/SIGCOMM '99* (September 1999).

[3] Carlson, J. M., and Doyle, J. Highly optimized tolerance: A mechanism for power laws in designed systems. *Physical Review E 60* (1999), 1412–1427.

[4] Chiu, D.M., and Jain, R. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems 17* (June 1989).

[5] Crovella, M. E., and Bestavros, A. Self-Similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking 5*, 6 (December 1997), 835–846.

[6] Downey, A. B. The structural cause of file size distributions. Tech. rep., Computer Science Department, Wellesley College, To appear at MASCOTS '01 2001.

[7] Fall, K., and Varadhan, K. *NS notes and documentation.* UC Berkeley, LBL, USC/ISI and Xerox PARC, Oct 2001.

[8] Floyd, S., Gummadi, R., and Shenker, Scott. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. Tech. rep., AT&T Center for Internet Research at ICSI, August 2001.

[9] Floyd, S., and Jacobson, V. Random Early Detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking 1*, 4 (August 1993), 397–413.

[10] Hollot, C.V., Misra, V., Towsley, D., and Gong, W.B. A control theoretic analysis of RED. In *Proceedings of IEEE/INFOCOM* (2001).

[11] Hollot, C.V., Misra, V., Towsley, D., and Gong, W.B. On designing improved controllers for AQM routers supporting TCP flows. In *Proceedings of IEEE/INFOCOM* (2001).

[12] Kunniyur, S., and Srikant, R. Analysis and design of an adaptive virtual queue algorithm for active queue management. In *Proceedings of ACM/SIGCOMM '2001* (2001).

[13] Liu, Y., Gong, W.B., and Shenoy, P. On the impact of concurrent downloads. In *Proceedings of 2001 Winter Simulation Conference* (December 2001), pp. 1300–1305.

[14] Misra, V., Gong, W.B., and Towsley, D. Fluid based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of ACM/SIGCOMM* (2000).

[15] Morris, R. TCP behavior with many flows. In *Proceedings of IEEE Conference on Network Protocols* (1997).

[16] Unknown. *FlashGet Web Page (http://www.amazesoft.com/index.htm).* Amazesoft Company, 2001.

[17] Zhang, H., Hollot, C.V., Towsley, D., and Misra, V. A self-tuning structure for adaptation in tcp/aqm networks. Tech. rep., Department of Computer Sciences, UMass, Amherst, July 2002.