Rate-Adjustment Algorithm for Aggregate TCP Congestion-Control

Peerapol Tinnakornsrisuphap †, Rajeev Agrawal †, Wu-chun Feng ‡
tinnakor@cae.wisc.edu, agrawal@engr.wisc.edu, feng@lanl.gov

† Department of Electrical and Computer Engineering
University of Wisconsin-Madison
1415 Engineering Drive
Madison, WI 53706

‡ RADIANT group
P.O. Box 1663, M.S. B255
Los Alamos National Laboratory
Los Alamos, NM 87545

Abstract— **The TCP congestion-control mechanism is an algorithm designed to probe the available bandwidth of the network path that TCP packets traverse. However, it is well-known that the TCP congestion-control mechanism does not perform well on networks with a large bandwidth-delay product due to the slow dynamics in adapting its congestion window, especially for short-lived flows. One promising solution to the problem is to aggregate and share the path information among TCP connections that traverse the same bottleneck path, i.e., Aggregate TCP. However, this paper shows via a queueing analysis of a generalized processor-sharing (GPS) queue with regularly-varying service time that a simple aggregation of local TCP connections together into a single aggregate TCP connection can result in a severe performance degradation. To prevent such a degradation, we introduce a rate-adjustment algorithm. Our simulation confirms that by utilizing our rate-adjustment algorithm on aggregate TCP, connections which would normally receive poor service achieve significant performance improvements without penalizing connections which already receive good service.**

Keywords— **TCP Congestion-Control, Aggregation and State Sharing, Rate-Adjustment Algorithm, GPS, Regularly-varying Service Time.**

I. INTRODUCTION

IN the current Internet architecture, there are several reasons that contribute to an increase in the number of simultaneous TCP connections between end-to-end destinations. First of all, in current TCP implementations, different applications connecting between the same end-to-end hosts will utilize separate TCP connections. Second, in the current Internet architecture, end-to-end hosts with more TCP connections will obtain a larger portion of the bottleneck bandwidth (See Appendix A of [1]), providing an incentive to increase the number of TCP connections even further. Lastly, TCP has been shown to perform poorly over networks with a large bandwidth-delay product because the adaptation of its congestion is too slow [2]. By breaking a single TCP connection into several concurrent TCP connections, a user can increase his performance at the cost of performance degradation of other customers. By all these reasons, it is reasonable to expect that the number of simultaneous TCP connections will keep increasing, and may lead to persistent congestion if there is no modification in how to connect end-to-end hosts over the Internet by TCP.

Many researchers have addressed these problems by proposing an aggregate TCP. An aggregate TCP would set-up a universal TCP connection which multiplexes local TCP connections into a single, persistent TCP connection. There are many variants of the aggregate TCP idea, each of which multiplexes local traffic in different level. For example, *Aggregate TCP* [3] multiplexes traffic with the same end-to-end subnetworks into a single TCP connection. *TCP Trunking* [4] performs a per-path congestion control in which all TCP connections between two sites, which can be any subset of a path TCP packets traverse on, are aggregated together.

The problem with these aggregate TCP studies is that they seem to indicate by simulations that the performance of each individual transfer would be better or at least equivalent to when utilizing separate TCP connections. We will show later in Section 2 by a queueing analysis of a GPS queue with regularly-varying service times that by simply aggregating traffic together as described in previous studies can result in significantly worse performances than without aggregation for some customers under certain conditions. We then propose the *Integrated TCP (I-TCP)* which combines an aggregate TCP with a rate-adjustment algorithm to prevent such conditions from occurring. We later show that the overall performance after utilizing I-

This research was supported by funding from the Los Alamos Computer Sciences Institute at Los Alamos National Laboratory.

TCP is significantly better than a simple aggregate TCP.

The rest of this paper is organized as follows. In Section 2, issues in the current aggregate and non-aggregate TCP ideas are discussed. Section 3 presents the rate-adjustment algorithm, and then follows by the protocol specification. A simulation study and analysis of the proposed protocol are presented in Section 4. Finally, Section 5 concludes the paper and offers suggestions for future work.

II. Motivation

Before we introduce the technique for the rate-adjustment algorithm, problems with TCP and aggregate TCP are demonstrated.

In this section, we introduce the following notations. For any two real functions $g(.)$ and $h(.)$, let $g(x) \sim h(x)$ denote $\lim_{x\to\infty} g(x)/h(x) = 1$. And for any non-negative random variable $\mathbf{X}$ with finite mean and distribution function $F(.)$, let $F^r(x) = \frac{1}{\mathcal{E}\mathbf{X}} \int_0^x (1-F(y))dy$ be the distribution of random variable $\mathbf{X}^\mathbf{r}$, where $\mathbf{X}^\mathbf{r}$ is the residual lifetime of $\mathbf{X}$. Denote by $\mathbf{W}_i, \mathbf{B}_i, \mathbf{S}_i, \mathbf{V}_i$ the random variable associated with the busy period distribution, the service time of the customers, the sojourn time and the stationary workload distribution in queue i respectively. The classes of regularly-varying and intermediately regularly-varying distributions are denoted with $\mathcal{R}, \mathcal{IR}$. The definitions of these classes can be found in Appendix A of [5].

Currently, all hosts transmitting files through the same bottleneck link utilize separate TCP connections for each file, with the exception of HTTP1.1 traffic that multiplexes all files in a web page into a single TCP connection which can also be treated as a large, single file transfer. If all connections utilize TCP, then each connection will get approximately $1/N$ portion of the bottleneck bandwidth if there are N active connections, assuming all connections have equal round-trip delay. Assume the arrival of each connection being a Poisson process, and host i submits a file transfer request with the file size having a regularly-varying distribution of index $-\nu_i$. Then the behavior at the bottleneck link can be described as a M/G/1 Processor Sharing queue with multiclass customers. The reason we choose a regularly-varying file size distribution is from the findings that the distribution of filesizes and the length of web sessions on the Internet is heavy-tailed [6], [7]. In this queue, we are interested in the tail of sojourn time distribution, which represents the time required to successfully transmit a file of a customer in class i. In multiclass M/G/1 Processor Sharing queue, we have the following relationship:

Theorem 1 (Zwart [8]) Suppose customers in all classes processor-share a single server. For $i = 1, ..., N$, and non-integer $\nu_i > 1$, $\mathcal{P}[\mathbf{B}_i > x]$ is regularly-varying of index $-\nu_i$ if and only if $\mathcal{P}[\mathbf{S}_i > x]$ is regularly-varying of index $-\nu_i$. Both imply that

$$\mathcal{P}[\mathbf{S}_i > x] \sim \mathcal{P}[\mathbf{B}_i > (1-\rho)x], \quad x \to \infty, \tag{1}$$

where $\rho < 1$ represents the average load in the queue.

In other words, if the input traffic does not overload the router, then the time required to transmit a file is proportional to filesize. The problem with this scheme on the Internet is, however, customers have no information if the network is overloaded and there is no incentive to discourage customers from submitting more jobs, which ultimately can overload the available bandwidth and penalize customers who are well-behaved. Because if $\rho > 1$, the queue will become unstable and all customers will experience increasingly longer sojourn time in return.

A simple aggregation of all requests in a class into a single TCP connection can also introduce a new problem, however. Although it seems initially that doing so will provide separation and protection of customers in each class, it can be shown that an aggregate connection may be induced by other connections to have significantly worse service under certain conditions. Suppose we have a fixed number of subnetworks (N) sharing the same bottleneck link. If each subnetwork aggregates data into a single TCP connection, then each aggregate TCP connection will get $1/i$ $(1 \le i \le N)$ portion of the bottleneck bandwidth where i is the number of active aggregate TCP connections. Therefore, this behavior at the bottleneck link can be approximated by a *generalized processor sharing (GPS)* queue with each connection having a weight of $1/N$ and hence a guaranteed rate of $1/N$. If aggregate gateways schedule packets to be transmitted by the aggregate TCP by the round-robin algorithm, then we can model the bandwidth available to each active customer as a queue where customers in each class processor-share the available bandwidth assigned by GPS. We are interested whether the sojourn time distribution of these customers are better with this scheme than in the separate TCP connections scheme. It is easy to see that even when the total load $\rho > 1$, the only classes of customers that will be unstable are those that transmit more than their guaranteed bandwidth, therefore offering partial protection for well-behaved connections from misbehaving connections. The more interesting case is when $\rho < 1$. Let ϕ_i be the guaranteed rate for the connection i, and ρ_i be the offered load of the connection i where $\rho = \sum_{i=1}^{N} \rho_i$.

First, we need to introduce a lemma.

Lemma 1: For $\phi_i > \rho_i$, the sojourn time distribution of customers who processor-share bandwidth of session i cannot be regularly-varying with different index than in the service time distribution.

Proof: This is a simple extension from Theorem 1. Since in a GPS queue, session i is guaranteed to receive $\phi_i > \rho_i$ portion of bandwidth, then by directly applying Theorem 1 we get the upper bound that is regularly-varying with index -ν_i. On the other hand, if the other sessions are all idle, then the full bandwidth is assigned to this session and by Theorem 1, the lower bound is also regularly-varying with the index -ν_i. Therefore, the sojourn time distribution cannot be regularly-varying with a different index. ■

It is worth noting that this does not imply that the sojourn time distribution of the customer in session i will be regularly-varying with index $-\nu_i$. In fact, it may not have regularly-varying distribution at all, depending on the behavior of the other sessions. However, since the sojourn time distribution is bounded by two regularly-varying distributions with the same index as the service time distribution, the time required to complete the service of each customer is not significantly different than in the previous scheme.

The condition $\rho_i < \phi_i$ is a sufficient but not necessary condition for source i to be stable (or the workload in the queue does not increase indefinitely). Borst et al. [9] show that even for $\rho > 1$ and $\rho_i > \phi_i$, source i can still be stable within certain conditions. The more interesting question is whether the global stability condition ($\rho < 1$) would be enough to guarantee no degradation in the performance of all customers. Borst et al. demonstrate that under a certain condition, *induced burstiness* can occur as in the following theorem.

Theorem 2 (Borst et al. [5]) Assume that the arrival process for class 2 customers is Poisson, and the service discipline is GPS. If $\rho_1 > \phi_1$, $\rho_1 + \rho_2 < 1$, $\mathbf{B}_2^r(.) \in \mathcal{IR}$, and $\mathcal{P}[\mathbf{B}_1^r > x] = o(\mathcal{P}[\mathbf{B}_2^r > x])$ as $x \to \infty$, then

$$\mathcal{P}[\mathbf{V}_1 > x] \sim \frac{\phi_2 - \rho_2}{\phi_2} \frac{\rho_2}{1 - \rho_1 - \rho_2} \mathcal{P}[\mathbf{W}_2^r > x/(\rho_1 - \phi_1)] \quad (2)$$

where $\mathbf{W}_2^r$ can be obtained from Theorem 4 by letting $c = \phi_2$ and $\rho = \rho_2$.

Or in other words, improper rate assignments might cause the workload in a source to be induced to be as bursty as the residual busy period of the other queue. This scenario can be extended to the many sources case although the asymptotic workload distribution might be extremely difficult to derive. In the following theorem, we show that not only the workload distribution is affected, but the sojourn distribution of customers who processor-share this session can also be induced to be burstier.

Theorem 3 (Lower bound) Assume that the customers arrival are Poisson processes, $\rho_1 > \phi_1$, $\rho_1 + \rho_2 < 1$, and $\mathbf{B_i}$, $i = 1, 2$ are regularly-varying with indices $-\nu_2 > -\nu_1$. If the service discipline in queue 1 is processor sharing, then $\forall \delta > 0, \mathcal{P}[\mathbf{S_1} > t] \geq l(t)$ where $l(t)$ is a regularly-varying function with index $-\nu_2 + 1 - \delta$.

Proof: The proof of this Theorem is stated in Appendix A. ■

As a consequence of Lemma 1 and Theorem 3, if the total offered load is enough for the queue to be stable ($\rho < 1$), we can lower the guaranteed rate of queue 2 and increase the guaranteed rate for queue 1, so that the tail of the sojourn time distributions of both connections are of the same indices as the tail of their respective service time distributions. In doing so, customers utilizing queue 2 will receive no significant service degradation while the customers in queue 1 would receive significantly better service time. This idea can be generalized to a scenario where there are more than two queues, in which if queues with guaranteed service rate higher than offered load lower their guaranteed service rates, then customers in other connections would receive significantly better service without degrading customers that are already receiving good service.

Therefore, it is very important for an aggregate TCP connection to come up with an algorithm to adjust its service rate cooperatively with the other connections. The difficulties of doing this on the Internet arise from the following reasons. First, the architecture of the Internet is *decentralized* such that each connection has no knowledge of the number or the status of the other connections utilizing the same bottleneck link. So in adjusting the rate, this aggregate TCP connections need to work only on the information available locally (such as queue length in the aggregate TCP buffer, round-trip delay or packet losses). Second, the offered load from users and the network conditions are always changing with time, with the change in each component at different time scales. Last, the algorithm needs to take into account the existing TCPs in the Internet. This new algorithm should neither penalize nor favor the previous implementations of TCP while the rate adjustment algorithm should help improve the performance when working with similar aggregate TCP.

In the case where $\rho > 1$, aggregate TCP connections should try to adjust their weight to the lowest possible that is still greater than their offered load. By doing this, they free up a portion of bandwidth which might help some connections that would become unstable without help. Although with $\rho > 1$, it is unavoidable that some connections would face instability.

III. Rate-Adjustment Algorithm

First, we need to establish a relationship between the portion of bandwidth a TCP connection receives and the parameter settings in the TCP congestion-control algorithm. The TCP implementation we based our design on is TCP Vegas [10], although we could extend a similar idea to TCP Reno [11] with minor adjustments.

Following the analysis by Bonald [12], we found by fluid approximation that if we let $\alpha = \beta$, where α and β are the parameters of TCP Vegas, then all TCP Vegas connections will get approximately $1/N$ fraction of the bottleneck bandwidth when N is the number of the active connections. We introduce a new parameter $\kappa_i \in \Re^+$ as the weight of connection i. It is easy to show from [12] that when connection i adapts its congestion window similar to the sum of κ_i TCP Vegas connections (increasing/decreasing congestion window κ_i packets per round-trip, and setting $\alpha = \beta = \kappa_i$), then the connection i will get

$$\phi_i = \frac{\kappa_i}{\sum_{j=1}^{N} \kappa_j} \tag{3}$$

fraction of bandwidth where N is the number of active connections. The trace from a simulation demonstrating such behavior is shown in Figure 1.

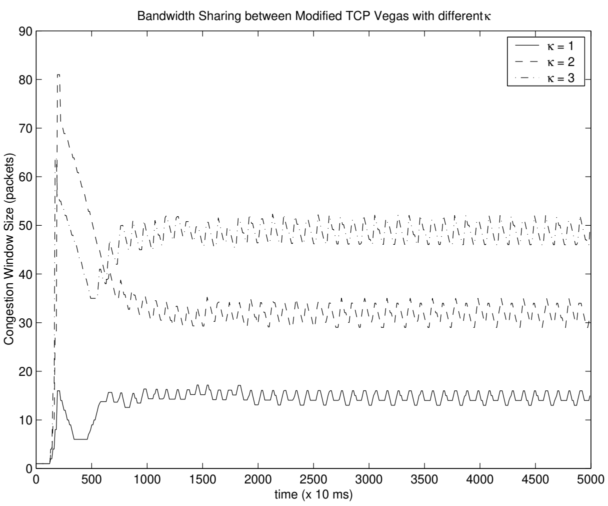

Fig. 1. Sample Congestion Window evolution of Modified TCP Vegas Connections with different κ.

From this argument, we can relate the portion of bandwidth each connection is getting as the function of $(\kappa_1, \kappa_2, ..., \kappa_N)$. As stated earlier, our objective is to adapt the parameter κ_i such that $\phi_i > \rho_i$ for all i when the total offered load is less than the bottleneck link capacity. The range of κ_i needs to be limited to $1 \leq \kappa_i \leq M_i$, where M_i is the number of active local connections utilizing aggregate connection i. We need to limit the range of κ_i so that I-TCP would not behave more aggressively than when we utilize separate TCP connections and penalize any connections using other implementations of TCP.

A rigorous analysis of this system is very complicated as various aspects of the system are all coupled together. Thus we assume that the dynamics of each mechanism are of different timescales. Given two mechanisms a and b, we denote $a << b$ as the case where mechanism a settles to a steady state much faster than mechanism b. We assume that network dynamics $<<$ congestion window dynamics $<<$ offered load dynamics $<<$ connection-level dynamics. For example, when analyzing the bandwidth each TCP connection receives after its congestion window has changed, we can analyze each connection as if it got a new share of bandwidth immediately after its congestion window changed. With the separation of timescales, we can 'search' for the appropriate value of κ while treating the offered load (ρ) as a constant. Then the problem in two dimensions is, in fact, a line search problem shown in Figure 2.

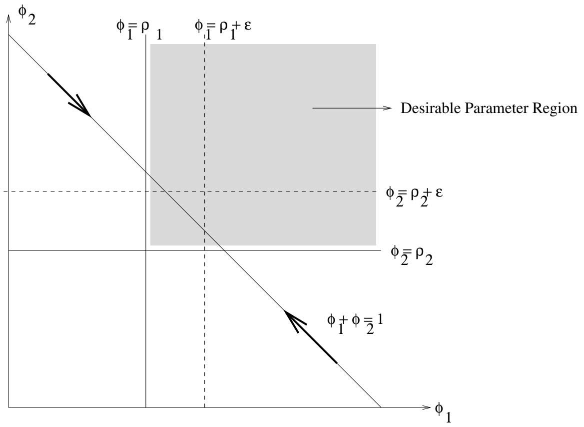

Fig. 2. Two-dimensional Line Search Problem for the appropriate values of κ.

A. *Protocol Specification*

We use the queue length in the aggregate TCP buffer as an indication of the difference between ϕ_j and ρ_j. Suppose $\rho_j > \phi_j$ and all other connections fully utilize their given bandwidth, then the queue will grow approximately linearly as a function of time with slope $\rho_j - \phi_j$. The goal of the rate-adjustment algorithm is to use this information to increase ϕ_j into the desirable parameter region. There are two alternatives to achieve that goal : (1) increase κ_j, or (2) decrease κ_i, for any $i \neq j$. Since each connection has to work asynchronously and without knowledge of other connections, each I-TCP adjusts its κ by an additive increase/multiplicative decrease scheme similar to what TCP Reno used to hunt for available bandwidth. If the slope of queue length is positive, then $\kappa_i = \kappa_i + \Delta\kappa_i$. If the slope of queue length is negative, then I-TCP does a multiplicative decrease or $\kappa_i = (1 - \gamma)\kappa_i$.

The multiplicative decrease part of the rate-adjustment algorithm is more simple than the additive increase part. Usually, we prefer to increase κ more often than to decrease κ since higher value of κ will enable the congestion window to converge to the steady state much faster than a low value of κ. Therefore, to make decreasing κ more difficult and less often than other connections increase their κ, we decrease κ by a factor $0 < \gamma < 1$ only when the slope of the queue length is negative in two consecutive rounds. Because if other connections increase κ before that, ϕ for the connections that have $\phi_i > \rho_i$ would be decreased and if it decreases too much, the slope of queue length might turn positive in the other round. By this mechanism, we can prevent oscillation in the value of κ to a certain degree.

One major difference of our algorithm compared to the additive increase/multiplicative decrease algorithm in TCP Reno is that the additive increase size ($\Delta\kappa$) is not constant and can be reasonably approximated to enable faster convergence. Now suppose that we can accurately approximate the slope of queue growth in queue j (L_j), then

$$L_j \approx \rho_j - \phi_j = \rho_j - \frac{\kappa_j}{\sum_{i=1}^N \kappa_i} \quad (4)$$

We need to find the value $\Delta\kappa_j$ that makes $\phi_j = \rho_j + \epsilon$ where ϵ is a parameter in this algorithm and its importance will be discussed in details later. So we have

$$\rho_j + \epsilon - \frac{\kappa_j + \Delta\kappa_j}{\sum_{i=1}^N \kappa_i + \Delta\kappa_j} = 0 \quad (5)$$

From (4) and (5), we get

$$\Delta\kappa_j = \frac{(L_j + \epsilon)}{1 - (\rho_j + \epsilon)}(\sum_{i=1}^N \kappa_i) > (L_j + \epsilon)(\sum_{i=1}^N \kappa_i) \quad (6)$$

In the Internet, we can approximate the value of L by the slope of queue growth divided by the bottleneck bandwidth, or

$$L_j \approx \frac{\text{slope of queue growth}}{\text{Bottleneck Bandwidth}} \quad (7)$$

Typically the value of bottleneck bandwidth is not known to a TCP connection and we could approximate it by the expected bandwidth in TCP Vegas with parameter κ_j with the following relationship :

$$\frac{\kappa_j}{\sum_{i=1}^N \kappa_i}\text{Bottleneck Bandwidth} \approx \text{Expected Bandwidth}$$
$$= \frac{cwnd}{\text{minimum RTT}} \quad (8)$$

where $cwnd$ is the size of congestion window.

Let

$$L'_j = \frac{\text{(slope of queue growth)(minimum RTT)}}{cwnd} \quad (9)$$

Or $L'_j \approx \frac{\sum_{i=1}^N \kappa_i}{\kappa_j} L_j$. Now define

$$\Delta\kappa'_j := 0.5(L'_j + \epsilon)\kappa_j = 0.5(L_j \sum_{i=1}^N \kappa_i + \epsilon\kappa_j)$$
$$< (L_j + \epsilon)(\sum_{i=1}^N \kappa_i) < \Delta\kappa_j \quad (10)$$

Note that $\Delta\kappa'_j$ uses only knowledge of κ_j, min RTT, $cwnd$, and slope of queue length which are all available locally and will be used as the value to add to κ_j in the additive increase phase. The factor 0.5 is used to preserve the inequality in the case of an imprecise approximation in (8) and would not have any impact in the proof of stability later.

As described earlier, I-TCP needs to calculate the slope of its queue growth. In practice, we can implement this by doing a linear regression on the sampled queue length. We need the following parameters : *no. of sample (C)* is the number of samples used to calculate the slope and *sampling period (P)* is the interval between each sample. Let $\hat{b}$ be the slope of queue length calculated by a linear regression on the sampled queue length and $\bar{y}$ be the average queue length. Then we can calculate L' in (9) by using $\hat{b}$ as the slope of queue growth and κ will be updated every PC seconds.

We can show under certain assumptions that this additive increase/multiplicative decrease algorithm will converge to $\phi_i > \rho_i,\ 1 \le i \le N$

Proposition 1: Consider the following assumptions : (i) only one connection adjusts its value of κ at a time, (ii) connections that have $\phi < \rho$ will always adjust their κ before connections that have $\phi > \rho$, (iii) if $\rho_i < \phi_i < \rho_i + \epsilon$, then κ_i would not be decreased, and (iv) κ_i would never have to decrease to 1.

Under the assumptions stated above, by using the additive increase/multiplicative decrease algorithm with the additive weight $\Delta\kappa'$ as in (10) and multiplicative decrease factor of $\gamma,\ 0 < \gamma < 1$, all the connections can achieve $\phi > \rho$ within finite time when $\sum_{i=1}^N \phi_i = 1, \sum_{i=1}^N(\rho_i + \epsilon) < 1$.

Proof: The proof of this proposition is given in Appendix B. ■

Alternatively, we can prove the convergence by using a system of differential equations. Suppose all connections continually update their κ. Then $\dot{x_i} = \rho_i - \frac{\kappa_i(t)}{\sum_{j=1}^N \kappa_j(t)}$

when $x_i(t) > 0$, and from $\Delta\kappa'$ in (10),

$$\dot{\kappa_i}(t) = \begin{cases} 0.5(\dot{x_i}(t)\sum_{j=1}^{N}\kappa_j(t) + \epsilon\kappa_i(t)) & \text{if } \dot{x_i} > 0 \\ -\gamma_i\kappa_i(t) & \text{otherwise} \end{cases} \tag{11}$$

which if we further simplify to additive increase/additive decrease, we have

$$\dot{\kappa_i}(t) = 0.5(\rho_i\sum_{j=1}^{N}\kappa_j(t) - \kappa_i(t) + \epsilon\kappa_i(t)) \tag{12}$$

This linear system has $0.5[\sum_{i=1}^{N}\rho_i + \epsilon - 1, \epsilon - 1, .., \epsilon - 1]$ as eigenvalues. Therefore, the system is stable if $\epsilon < 1 - \sum_{i=1}^{N}\rho_i$.

Since we do not have unlimited buffer space in the router, it is preferable to try to keep the average queue length below *max queue length* by introducing an additional parameter *high th*. If $\bar{y} > (high\ th)(max\ queue\ length)$ and $L' > 0$, then every time we update κ, we would increase κ at least one unit if the change does not cause κ to be out of range to try to get enough bandwidth to control the queue length to be below *max queue length* and would not decrease κ even when L' is negative.

Moreover, when the average queue length is very small, a small burst can cause an unrealistic value of L'. Therefore, we would increase κ only when the average queue length ($\bar{y}$) is greater than a certain threshold $(low\ th)(max\ queue\ length)$.

We need to introduce a random element into the time to update $\Delta\kappa$ to prevent synchronization between each I-TCP connection. Therefore, instead of updating $\Delta\kappa$ every PC seconds, we have a random waiting time determined by multiplying a uniform random variable between 0 and 1 by PC, after which we start sampling the queue length, and finally, $\Delta\kappa'$ is calculated and κ is updated after another PC seconds have passed.

B. Tradeoffs in parameters selection

The parameter ϵ, as mentioned in Proposition 1, ensures the convergence of the system when $\epsilon < \frac{1}{N}(1-\rho)$. On the other hand, a larger value of ϵ will speed up the convergence. Typically, it would be better to choose a conservative value of ϵ as shown later in the simulation; a small value of $\epsilon = 0.1$ suffices for the rate assignments to converge in reasonable time.

For connections to reduce their share of bandwidth, the parameter γ plays an important role. If γ is chosen to be very large, then the connections will suffer a severe performance loss before the additive increase mechanism reclaims the necessary portion of bandwidth back. However, small γ takes very long time before enough bandwidth is released for the other connections that need it. Especially in our scheme, a decrease in κ happens less frequently and is much harder. If there are many I-TCP connections sharing the same bottleneck link, however, it might not be as necessary for γ to be large as with a few connections because cumulative bandwidth releases from many connections can be substantial.

The most important parameters in practice might be the number of samples (C) and sampling period (P). In the earlier analysis, we assume we can perfectly estimate the value of L'. In practice, however, it is necessary for I-TCP to distinguish the increase in buffered packets between a burst from a new file transfer while ρ remains constant and a real trend where $\phi < \rho$. First, the sampling period needs to be large enough to let the network dynamics returns to steady state after a change in κ occurred. Then the number of samples need to be large enough to average out any temporary burst from a file. This value might have to be quite large if the average filesize is large comparing to the bottleneck bandwidth and might have to be even larger for *heavy-tailed* files because a file might be extremely large and if the number of samples is too small, this connection might be temporarily tricked into increasing its κ even when its current rate is good enough. It is obvious, however, that the large product of PC will result in a slow dynamics in adapting κ, which might result in networks operating in an inappropriate settings for a long time. But since the trend of bandwidth in the Internet is steadily increasing, this might be less of an issue in the future and κ-dynamics can be fairly fast.

Since the number of active connections utilizing an I-TCP connection can also be time-varying, its dynamics also plays a role in the overall performance. We have not considered the scenario where M varies with time here but our conjecture is that if the connection-level dynamics are slower than κ-dynamics, then we could still expect significant improvement in the performances by our algorithm.

The parameters *max queue length, low th, high th* are used together to specify the threshold where I-TCP will switch its operating mode. If the average queue length is smaller than $(low\ th)(max\ queue\ length)$ then I-TCP would not increase κ to prevent a small fluctuation in the buffer from causing an abrupt change in κ. In addition, I-TCP would not decrease κ if the average queue length exceeds $(high\ th)(max\ queue\ length)$ and will increase κ by at least one unit if L' is non-negative to try to force average queue length to be below $max\ queue\ length$. This would not guarantee that the average queue length would not exceed this value, however. It just indicates that I-TCP will try its best to do so. There are many options I-TCP can do when the average queue length exceeds this threshold. It might inform the local sources to slowdown, or simply does not return the ACK packets in the local TCP connections to force a timeout and wait until the average queue length in the aggregate gateway decrease below the threshold. In our simulations, we do not force any of these options but let the queue continue to grow as necessary to mimic a GPS queue as closely as possible.

IV. SIMULATIONS

In this section, we implement the protocol as described earlier and then show that simulation results agree with the theories presented in Section III. We later conclude with a discussion of the results.

A. Simulation Setup

We use *ns* [13], an event-driven simulator, as our simulation environment. Consider a client-server network consisting of one server and 10 clients in each of the two subnets. Each client is linked to an aggregate subnet gateway with a full-duplex link with speed 100 Mbps and delay 1 ms. Each subnet is then connected to a common router by link with speed 100 Mbps and delay 10 ms. This bottleneck router with speed 10 Mbps and buffer 10000 packets connects to the server with link delay 10 ms.

In each subnet, each client generates a file to transfer to the server with exponentially distributed interarrival time with mean 1 second. The filesize is determined by a Pareto random variable with mean {65000, 30000} bytes and index {-2.5, -1.2}. Each client establishes a local TCP Vegas connection to its aggregate gateway and transfer the file into the gateway's buffer. We modified the TCP Vegas implementation in ns to have the additional parameter κ as the weight of the connection and then use this modified TCP Vegas to transfer data in the gateway's buffer to the server. This κ is adapted according to the rate-adjustment algorithm described in the previous section. The throughput received by this modified TCP Vegas connection are being shared equally among each buffered file by the round-robin algorithm. In this study, we suppose there is no overhead in aggregating all streams together. The practical issues of aggregate TCP such as *split transaction* are discussed in details in [3].

In our first simulation, we simulate the induced burstiness in sojourn time distribution by two aggregate TCP as described in Theorem 3 by disabling our rate-adjustment mechanism (fix $\kappa = M = 10$) and letting $\rho_1 > 0.5$, $-\nu_1 < -\nu_2$, and $\rho_2 < 0.5$. The results are then compared to when the rate-adjustment is enabled. The parameters used for I-TCP is shown in Table I.

TABLE I
PARAMETERS IN THE TWO-SUBNET SIMULATION.

parameters	value
number of sample (C)	180
sampling period (P)	0.5 second
M	10
max_queue_length	200000 packets
ϵ	0.1
γ	0.2
low_th	0.2
$high\_th$	1.0

B. Simulation Results

Figure 3 shows the tail of sojourn time distribution of customers in Subnet 1 which have $\rho_1 = 0.52$. Since $M = 10$ in both I-TCP connections are equal, without rate adjustment, both connections have $\phi = 0.5$ and since the tail of service time of customers in Subnet 2 is heavier than in Subnet 1 ($\nu_2 < \nu_1$), customers in Subnet 1 will experience induced burstiness in their sojourn time distribution as shown in Theorem 3. The slope of the tail of sojourn time distribution without rate adjustment in Figure 3 is around -0.4, a slightly lighter than -0.2 predicted by Theorem 3. The reason that the tail dips significantly after 100 seconds is due to the finite time effect, or the events that is large comparing to the simulation time are less likely to occur than in the actual distribution. Anyway, it is evident that without rate adjustment, customers in Subnet 1 will receive extremely bad services which are unproportional to their service time. With rate-adjustment mechanism, the tail of sojourn time distribution of customers in Subnet 1 clearly has the slope as the service time distribution and receives significantly better service than without rate-adjustment.

It is expected that with rate-adjustment, sojourn times of customers in Subnet 2 with rate-adjustment will be greater than without rate-adjustment because I-TCP of Subnet 2 has to give up a portion of its bandwidth to assist I-TCP of subnet 1. However, Figure 4 shows that the differences in sojourn time distributions in both case are not as significant as in Subnet 1 and both tails still have the same index as their service time distribution. Therefore, by giving up small portion of bandwidth, customers in Subnet 2 receives only a minimal degradation in their services while customers in Subnet 1 receives significantly better services. In other words, with rate adjustment, all customers receive a 'fair' service in a sense that the time required to finish a file transfer is proportional to its file size.

The convergence of rate assigned to both connections

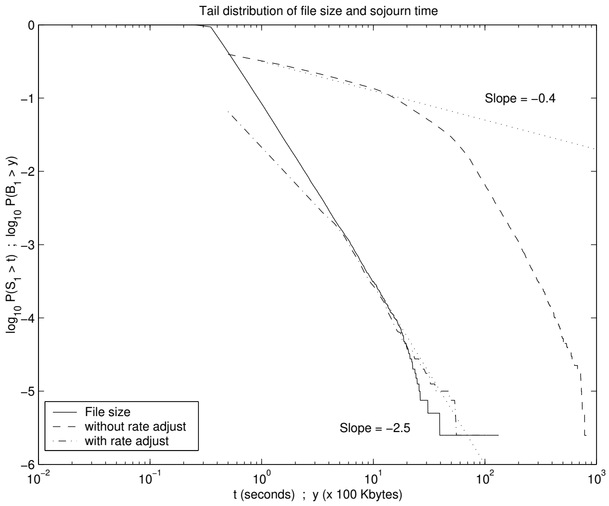

Fig. 3. Comparison of tail distribution between filesize and sojourn time of aggregate TCP with and without rate adjustment for clients in Subnet 1.

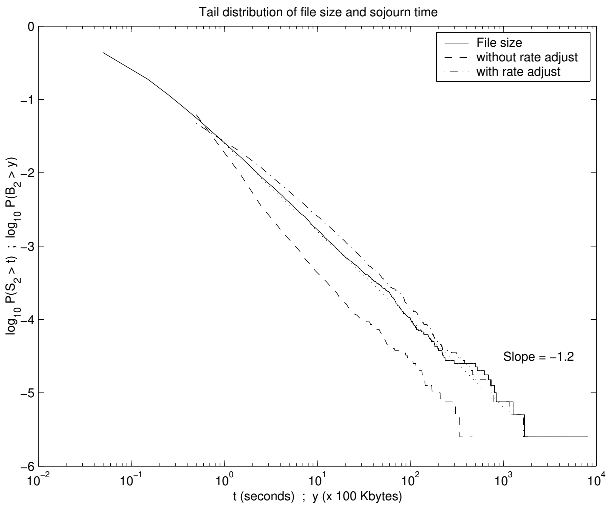

Fig. 4. Comparison of tail distribution between filesize and sojourn time of aggregate TCP with and without rate adjustment for clients in Subnet 2.

are shown in Figure 5. It shows that the guaranteed rates for both connections converge to a value $\phi > \rho$. It might take a long time before the rates finally converge but the more important issue is that I-TCP of Subnet 1 operates in the region where $\phi_1 > \rho_1$ almost from the start and therefore preventing induced burstiness from happening.

C. Multiple I-TCP connections simulation

In the previous section, we only simulated two I-TCP connections. Similar settings will be simulated but the number of I-TCP connections will be increased to four to see whether the convergence in rate assignment still holds or not. Each client in each subnet generates files with index {-2.5, -1.2, -1.7, -2.2} and mean

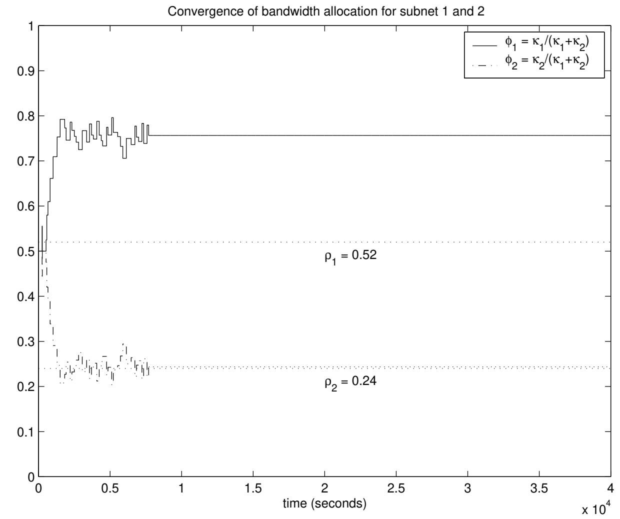

Fig. 5. Bandwidth allocation for subnet 1 and 2.

{35000,10000,10000,30000} bytes with interarrival times of one second. However, we decrease the number of sample (C) to 120 samples.

Figure 6 shows that the the rate assignment for all the connections converge to the region where $\phi > \rho$. One interesting note is that we use a smaller sampling window (120 samples) than in the previous simulation. This is because the mean file size in this simulation is smaller than in the previous simulation, enabling us to smooth the average load in smaller amount of time.

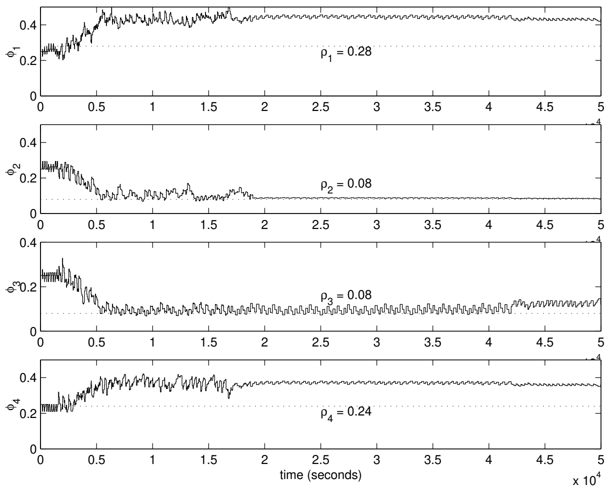

Fig. 6. Bandwidth allocation for the four-subnets simulation.

V. Conclusions

In this study, we have demonstrated the drawbacks of the simple aggregate TCP idea. More specifically, an aggregate TCP connection can be induced to be more bursty by other sources which have heavier tailed service times and the performance of the aggregate TCP connection will be severely degraded in this scenario, comparing to when each file utilizes its own TCP connection. We then propose the rate-adjustment algorithm which approximates the difference between the available bandwidth and the offered load of each aggregate TCP connection and then adapts accordingly. The connections that have excess bandwidth will give up a portion of its bandwidth to other connections without causing a significant performance loss for its own customers. We prove that, under a separation of timescale assumption, aggregate TCP with rate adjustment (or Integrated TCP) adapts its transmission in such a way that prevents induced burstiness from occurring. The simulation results also support the theoretical findings in this study.

One might question whether the proposed Integrated TCP would be useful in practical environments, where the environments are much more diverse than what we have considered in this study. For example, the network path that I-TCP connections traverse might change during the long lifetime of such connections. Or the number of local TCP connections can be changing with time. The policy to admit or terminate a local connection from an Integrated TCP connection would have an impact on the overall performance. Therefore, further studies involving these dynamics are still needed.

Although our proof of induced burstiness in the sojourn time distribution assumes that the distributions of filesize are regularly-varying. We believe that an extension to a wider class of distributions is possible. And since our rate-adjustment algorithm does not assume anything about the filesize distribution, we believe it can still be used to prevent induced burstiness in these cases. Another interesting future work is to investigate the parameters setting in I-TCP and find a methodology in choosing the optimal values for the algorithm. For example, it is preferable to accurately approximate the queue growth in the smallest sample size possible to enable faster convergence.

Future works will also include the performance of I-TCP in the presence of TCP Reno or other implementations of TCP. In many studies [12], [14], TCP Vegas is shown to perform poorly with the presence of TCP Reno. By utilizing RED gateway, it is believed that TCP Vegas would be more competitive to TCP Reno connection. So it might be interesting to see how I-TCP performs with RED as the buffer management scheme. Furthermore, it would be interesting to compare the performance of I-TCP with separate TCP connections. If the results turn out to be in favor of I-TCP, it would add an incentive for the deployments of I-TCP in the Internet.

ACKNOWLEDGMENTS

We gratefully acknowledge helpful discussions and suggestions from Thomas G. Kurtz and Armand Makowski.

REFERENCES

[1] Sally Floyd and Kevin Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, Aug. 1999.

[2] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Transactions on Networking*, June 1997.

[3] Prashant Pradhan, Tzi cker Chiueh, and Anindya Neogi, "Aggregate TCP congestion control using multiple network probing," in *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS2000)*, Taiwan, April 2000.

[4] H. T. Kung and S. Y. Wang, "TCP Trunking: Design, implementation and performance," in *Proceedings of the 7th International Conference on Network Protocols (ICNP'99)*, Oct. 1999, pp. 222–231.

[5] Sem Borst, Onno J. Boxma, and Pedrag Jelenkovic, "Induced burstiness in generalized processor sharing queues with long-tailed traffic flows," in *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, Sept. 1999.

[6] Vern Paxson and Sally Floyd, "Wide area traffic: The failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 226–244, 1995.

[7] Mark Crovella and Azer Bestavros, "Explaining World Wide Web traffic self-similarity," Tech. Rep. TR-95-015, Computer Science Department, Boston University, 1995.

[8] A. P. Zwart, "Sojourn times in a multiclass processor sharing queue," Tech. Rep. COSOR98-22, Eindhoven University of Technology, 1998.

[9] Sem Borst, Onno J. Boxma, and Pedrag Jelenkovic, "Asymptotic behavior of generalized processor sharing with long-tailed traffic sources," in *Proceedings of INFOCOM'2000*, Tel-Aviv, Israel, Mar. 2000.

[10] Lawrence S. Brakmo and Larry L. Peterson, "TCP Vegas: end to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[11] Van Jacobson, "Congestion avoidance and control," in *Proceedings of SIGCOMM'88 Symposium*, Aug. 1988, pp. 314–332.

[12] Thomas Bonald, "Comparison of TCP Reno and TCP Vegas via fluid approximation," Tech. Rep. 3563, INRIA, Nov. 1998.

[13] "UCB/VINT/LBNL Network Simulator (ns) version 2.0," http://www-mash.cs.berkeley.edu/ns.

[14] Jeonghoon Mo, Richard J. La, Venkat Anantharam, and Jean Walrand, "Analysis and comparison of TCP Reno and Vegas," in *Proceedings of INFOCOM'99*, 1999.

[15] A. Jean-Marie and P. Robert, "On the transient behavior of the processor sharing queue," *Queueing Systems*, vol. 17, pp. 129–136, 1994.

APPENDIX

I. PROOF OF THEOREM 3

First, we need the following theorems.

Theorem 4 (Borst et al. [5]) If the arrival process in to a queue which is serving at constant rate c is Poisson, $\mathbf{B}^{\mathbf{r}} \in$

$\mathcal{IR}$, and $\rho < c$, then

$$\mathcal{P}[\mathbf{W}^r > x] \sim \frac{c}{c-\rho}\mathcal{P}[\mathbf{B}^r > x(c-\rho)]$$

Theorem 5 (Jean-Marie and Robert [15]) If L_t is the number of customers in the transient processor-sharing queue at time t with $\rho > 1$, then

$$\lim_{t\to+\infty} \frac{L_t}{t} = \alpha, \quad a.s.$$

where α is a constant, non-negative solution of the following equation

$$\alpha = \lambda(1 - \int e^{-\alpha y} F(dy)) \tag{13}$$

and F is the distribution of the service time of customers.

Before giving the formal proof of Theorem 3, we first provide an intuitive interpretation. When queue 2 is busy, queue 1 only serves at rate ϕ_1, while it receives service requests at rate $\rho_1 > \phi_1$. Therefore, the number of customers in queue 1 grows linearly asymptotically as shown in Theorem 5. This growing number of customers results in longer completion time of each customer who processor-shares this queue. The number of customers will keep growing until queue 2 becomes idle again, at which point queue 1 will start finishing work from the accumulated customers. Therefore, the tail of sojourn time distribution of customers arriving in queue 1 while queue 2 is busy will behave asymptotically at least as heavy as the residual busy period of queue 2, which is one index heavier than the service time of queue 2.

Proof: First we introduce a modified system (MS) which has the following behaviors: (i) Queue 2 is always served at rate 1 (ii) Queue 1 is served at rate 1 when queue 2 is empty, and is served at rate ϕ_1 when queue 2 is busy. It is easy to see that for the same arrival process in the modified queue 2, $\{t : W_2(t) > 0\} \supset \{t : W_2^{MS}(t) > 0\}$. Therefore in the modified system, queue 1 is being drained at the rate equal or faster than in the original system, resulting in smaller sojourn time for each customer. As a consequence, $\mathcal{P}(\mathbf{S_1} > x) \geq \mathcal{P}(\mathbf{S_1^{MS}} > x)$. In the modified system, with probability $\lambda_2 < 1$, a tagged customer in queue 1 arrived while queue 2 is busy. Then the service rate of queue 1 will be ϕ_1 and the remaining busy period of queue 2 will be $\mathbf{W_2^r}$ by PASTA property. The sojourn time for this customer is bounded from below by the sojourn time of a customer in queue 1 that arrives at time 0 while queue 1 is empty and queue 2 is busy. Let $\mathbf{L_1}(s)$ be the number of customers in queue 1 at time s. Then we have

$$\mathcal{P}[\mathbf{S_1} > t] \geq \lambda_2 \mathcal{P}[\mathbf{W_2^r} > t, \mathbf{B_1} \geq \int_0^t \frac{\phi_1}{\mathbf{L_1}(s)} ds] \tag{14}$$

Let $\mathbf{L_1'}(t)$ be the number of the customers in queue 1 starting with zero customer, having arrival process similar to the arrival process after the tagged customer arrived and being served at rate ϕ_1. Then (14) becomes

$$\begin{aligned}\mathcal{P}[\mathbf{S_1} > t] &\geq \lambda_2 \mathcal{P}[\mathbf{W_2^r} > t, \mathbf{B_1} > \int_0^t \frac{\phi_1}{1+\mathbf{L_1'}(s)} ds] \\ &\geq \lambda_2 \mathcal{P}[\mathbf{W_2^r} > t, \mathbf{B_1} > \phi_1 t_0 + \int_{t_0}^t \frac{\phi_1}{1+(\alpha-\epsilon)s} ds, \\ &\qquad \mathbf{L_1'}(s) > (\alpha-\epsilon)s; \forall s > t_0]\end{aligned} \tag{15}$$

Since $\mathbf{W_2^r}, \mathbf{B_1}, \mathbf{L_1'}$ are independent in the modified system, then

$$\begin{aligned}\mathcal{P}[\mathbf{S_1} > t] &\geq \lambda_2 \mathcal{P}[\mathbf{B_1} > \phi_1 t_0 + \int_{t_0}^t \frac{\phi_1}{1+(\alpha-\epsilon)s} ds] \\ &\qquad \mathcal{P}[\mathbf{W_2^r} > t]\mathcal{P}[\mathbf{L_1'}(s) > (\alpha-\epsilon)s; \forall s > t_0] \quad (16) \\ &= \lambda_2 \mathcal{P}[\mathbf{B_1} > \phi_1 t_0 + \frac{\phi_1}{(\alpha-\epsilon)} \ln \frac{1+(\alpha-\epsilon)t}{1+(\alpha-\epsilon)t_0}] \\ &\qquad \mathcal{P}[\mathbf{W_2^r} > t]\mathcal{P}[\mathbf{L_1'}(s) > (\alpha-\epsilon)s; \forall s > t_0] \quad (17)\end{aligned}$$

From Theorem 4, $\mathcal{P}[\mathbf{W_2^r} > t] \sim \frac{1}{1-\rho}\mathcal{P}[\mathbf{B_2^r} > t(1-\rho)] \sim \frac{t^{-\nu_2+1}}{\nu_2(1-\rho)^{\nu_2+2}}$. And from Theorem 5, given any $\epsilon, \zeta > 0$, $\exists t_0$ such that $\mathcal{P}[\mathbf{L_1'}(s) \geq (\alpha-\epsilon)s; \forall s > t_0] \geq 1-\zeta$, where α satisfies the equation $\alpha = \lambda_1(\phi_1 - \int e^{-\alpha y} F_{\mathbf{B_1}}(dy))$. Therefore,

$$\begin{aligned}&\liminf_{t\to\infty} \mathcal{P}[\mathbf{S_1} > t]/[\lambda_2(1-\zeta)(\frac{\phi_1}{\alpha-\epsilon})^{-\nu_1} \\ &\qquad (\ln\ (\alpha-\epsilon)t)^{-\nu_1} \frac{t^{-\nu_2+1}}{\nu_2(1-\rho)^{\nu_2+2}}] \geq 1\end{aligned} \tag{18}$$

which implies

$$\begin{aligned}&\liminf_{t\to\infty} \mathcal{P}[\mathbf{S_1} > t]/[\lambda_2(1-\zeta)(\frac{\phi_1}{\alpha-\epsilon})^{-\nu_1} \\ &\qquad \frac{t^{-\nu_2+1-\delta}}{\nu_2(1-\rho)^{\nu_2+2}}] \geq 1\end{aligned} \tag{19}$$

for any $\delta > 0$. ■

II. Proof of Proposition 1

To illustrate the problem, we first consider the simple $N = 2$ case and then extend the result to the more general case under the following assumptions : (i) only one connection adjusts its value of κ at a time (ii) connections that have $\phi < \rho$ will always adjust their κ before connections that have $\phi > \rho$ (iii) if $\rho_i < \phi_i < \rho_i + \epsilon$, then κ_i would never be decreased (iv) κ_i would never have to decrease to 1. Assumptions (i) and (ii) enable us to analyze the system as a synchronous system which is simpler to analyze. Assumption (iii) is used to force the convergence to be within finite time. And finally, Assumption (iv) is needed to prevent the technical difficulties which might arise for a very low value of ρ.

Suppose that initially, $\kappa_1/(\kappa_1+\kappa_2) = \phi_1 > \rho_1$, $\kappa_2/(\kappa_1+\kappa_2) = \phi_2 < \rho_2$ and $\rho_1+\rho_2 < 1$, then the desirable parameter region exists and we have the following lemmas.

Lemma 2: Assume $\phi_1 > \rho_1$ and $\phi_2 < \rho_2$, then by using $\Delta\kappa_2'$ as in (10), the new rate $\phi_1' > \rho_1$ and $\phi_2' < \rho_2 + \epsilon$ if $0 < \epsilon < 1-\rho_1-\rho_2$.

Proof: Since the upperbound for the value of ϕ_2 by using $\Delta\kappa_2'$ in this scheme is $\rho_2+\epsilon$, if $\epsilon < 1-\rho_1-\rho_2$, then $\phi_1' > 1-\rho_2-\epsilon > \rho_1$ or the increased value in κ_2 would not cause ϕ_1 to decrease smaller than ρ_1. ■

Lemma 3: Assume $\phi_1 = \kappa_1/(\kappa_1+\kappa_2) > \rho_1$ and $\phi_2 < \rho_2$. If $\exists a > 0$ such that $(\kappa_2+a)/(\kappa_1+\kappa_2+a) > \rho_2$ and $\kappa_2 + a < M_2$. Then by using $\Delta\kappa_2'$ as in (10), the rate assignment will converge to $\phi_1' > \rho_1$ and $\phi_2' > \rho_2$ in finite time if $0 < \epsilon < 1-\rho_1-\rho_2$.

Proof: Since $\Delta\kappa_2' > 0.5\epsilon\kappa_2 \geq 0.5\epsilon$, then if a exists, it will take at most $\lceil 2a/\epsilon \rceil$ updates to achieve $\phi_2' > \rho_2$. And from Lemma 2, we have $\phi_1' > \rho_1$, thereby achieving the desirable rate for both connections. ■

Lemma 4: Let ϕ_i^k be the guaranteed rate of connection i after the k^{th} update. If $\phi_1^k = \kappa_1/(\kappa_1+M_2) > \rho_1$, $\phi_2^k < \rho_2$ and $\phi_1^{k+1} = ((1-\gamma_1)\kappa_1)/((1-\gamma_1)\kappa_1+M_2) < \rho_1$, then $\phi_2^j > \rho_2$, $\forall j > k$. Furthermore, the rate assignment will converge to $\phi_1' > \rho_1$ and $\phi_2' > \rho_2$ in finite time if $0 < \epsilon < 1-\rho_1-\rho_2$.

Proof: Since $\phi_1^k = \kappa_1/(\kappa_1+M_2) > \rho_1$ and $\phi_1^{k+1} = ((1-\gamma_1)\kappa_1)/((1-\gamma_1)\kappa_1+M_2) < \rho_1$, then $\exists a : 0 < a < \gamma\kappa_1$ such that $((1-\gamma_1)\kappa_1+a)/((1-\gamma_1)\kappa_1+\kappa_2+a) > \rho_1$. Lemma 4 then follows directly from Lemmas 2 and 3. ■

By Lemmas 2, 3 and 4, we essentially prove convergence in the two-dimensional case, by using (10) to update the value of κ_i for $\epsilon < 1-\rho_1-\rho_2$. This idea can be extended to the case where the number of connections $N \geq 2$. Notice that now it is not always true that connections with $\phi_i > \rho_i$ will always remain in that region if other connections increase their κ. So we need to divide the connections into two groups: $\mathcal{A}^m = \{i : \phi_i^m > \rho_i^m\}$ and $\mathcal{B}^m = \{i : \phi_i^m \leq \rho_i^m\}$ represent the set of connections that has enough and not enough bandwidth after m^{th} update, respectively. Define $||\mathcal{A}^m|| = \sum_{i\in\mathcal{A}^m}(\phi_i^m - \rho_i^m)$ and $||\mathcal{B}^m|| = \sum_{i\in\mathcal{B}^m}(\rho_i^m - \phi_i^m)$ as the distance from the current rate assignment to the offered load of these two sets.

Lemma 5: For any initial conditions that satisfy the condition in Proposition 1, there exists a finite integer $m^* = \max\{l : \kappa_i^{j-1} \leq \kappa_i^j,\ 1 \leq i \leq N, 1 \leq j \leq l\}$ and $\forall j \in \mathcal{B}^{m^*}, \kappa_j^{m^*} = M_j$.

Proof: This lemma follows directly from the assumptions in Proposition 1. For any initial condition, if $i \in \mathcal{B}^l$ and $\kappa_i^l < M_i$ then there exists $l' > l$ such that $\kappa_i^l < \kappa_i^{l'} \leq M_i$. Let $\mathcal{C}^m = \{i \in \mathcal{B}^m : \kappa_i^m < M_i^m\} \subseteq \mathcal{B}^m$ be the set of connections that do not have enough rate and can still increase their κ. Since we assume that connections in $\mathcal{A}$ will not decrease their κ if there exists connections in $\mathcal{B}$ that still can increase their κ, then as long as $\mathcal{C}^m \neq \emptyset$, $\kappa_i^{m+1} \geq \kappa_i^m$, $\forall i$. And from $\Delta\kappa \geq 0.5\epsilon > 0$, then $\mathcal{C}^{m^*} = \emptyset$ for the first time at a finite integer m^*. ■

Lemma 6: For all $m < m^*$ in Lemma 5, either $||\mathcal{B}^{m+1}|| < ||\mathcal{B}^m||$ or $\mathcal{A}^{m+1} = \mathcal{A}^m \cup \{i\}$ where i is the connection that increases κ at the m^{th} update.

Proof: It is easy to see that if $\rho < 1$, $\mathcal{A}^m \neq \emptyset$ for any m. Then if connection i increases its κ_i at time m, then let $\Delta\phi_i = \phi_i^{m+1} - \phi_i^m$ be the change in the rate, which all other connections have to give up for the total of $\Delta\phi_i$. If $\phi_i^m + \Delta\phi_i > \rho_i$, then $\mathcal{A}^{m+1} = \mathcal{A}^m \cup \{i\}$. Otherwise, $i \in \mathcal{B}^{m+1}$ and since $\mathcal{A}^{m+1} \neq \emptyset$, then a portion δ, $0 < \delta < \Delta\phi_i$ must come from connections in $\mathcal{A}^{m+1}$ so $||\mathcal{A}^m|| - ||\mathcal{A}^{m+1}|| = \delta$ and since $||\mathcal{A}^m|| - ||\mathcal{B}^m|| = 1-\rho$ for any m, then $||\mathcal{B}^{m+1}|| < ||\mathcal{B}^m||$. ■

Lemma 7: Suppose connection i decreases its κ_i at time m and $i \in \mathcal{A}^m$, $\phi_i^m > \rho_i + \epsilon$ and $i \in \mathcal{B}^{m+1}$, then $\exists m' > m$, such that $\mathcal{A}^m \subseteq \mathcal{A}^{m'}$ and $||\mathcal{B}^m|| > ||\mathcal{B}^{m'}||$.

Proof: This condition indicates that $(1-\gamma)\kappa_i^m = \kappa_i^{m+1}$, $\phi_i^m > \phi_i^{m+1}$ and $\mathcal{C}^m = \emptyset$. Denote $\kappa_{A^m} = \sum_{l\in\mathcal{A}^m}\kappa_l$ and $\rho_{A^m} = \sum_{l\in\mathcal{A}^m}\rho_l$. Then we have $||\mathcal{B}^m|| = \rho_{B^m} - \kappa_{B^m}/(\kappa_{A^m}+\kappa_{B^m})$. Since $\mathcal{C}^{m+1} = \{i\}$, then at time $m+1$ only connection i can adapt its κ_i which will be increased until time $m+k$, $k \geq 2$ where $\kappa_i^{m+1} < \kappa_i^{m+k} < \kappa_i^m$ and $\rho_i \leq \phi_i^{m+k} \leq \rho_i + \epsilon$ (from Lemma 2). At $m+k$, $\phi_j^{m+k} > \phi_j^m$ for all $j \neq i$, therefore, $\mathcal{A}^m \subseteq \mathcal{A}^{m+k}$ and $||\mathcal{B}^m|| > ||\mathcal{B}^{m+k}||$. ■

Proof of Proposition 1: From Lemma 5, connections in $\mathcal{B}^0$ will increase their κ until either their κ hit the upper limit or their rates are enough. Then at time m^*, either $\mathcal{C}^{m^*} = \emptyset$ and $\mathcal{B}^{m^*} \neq \emptyset$ or $\mathcal{A}^{m^*} = 1, 2, ..., N$ by Lemmas 5 and 6. After m^*, connections from $\mathcal{A}$ will take turn decreasing their κ, where $||\mathcal{B}^m||$ is a strictly decreasing function of m except at time l when the condition in Lemma 7 is satisfied, i.e. a connection decreases its κ too much and it temporarily joins set $\mathcal{B}^{l+1}$. However, since this is the only connection in $\mathcal{C}^{l+1}$, it is the only connection that can adapt until l' as in Lemma 7, in which $||\mathcal{B}^{l'}|| < ||\mathcal{B}^l||$. For any $m > 0$ and $||\mathcal{B}^m|| > 0$, $\exists m' > m$ where $||\mathcal{B}^m|| > ||\mathcal{B}^{m'}||$. Therefore, $||B^m||$ will eventually converge to zero and will converge in finite time because we can show that $||\mathcal{B}^m|| - ||\mathcal{B}^{m'}||$ in Lemma 7 is bounded from below by $0.5\epsilon/(\sum_{i=1}^N M_i)^3 > 0$.