Heterogeneous Congestion Control: Efficiency, Fairness and Design

Ao Tang David Wei Steven H. Low
EAS Division, California Institute of Technology
Mung Chiang
EE Department, Princeton University

***Abstract*— When heterogeneous congestion control protocols that react to different pricing signals (e.g. packet loss, queueing delay, ECN marking etc.) share the same network, the current theory based on utility maximization fails to predict the network behavior. Unlike in a homogeneous network, the bandwidth allocation now depends on router parameters and flow arrival patterns. It can be non-unique, inefficient and unfair. This paper has two objectives. First, we demonstrate the intricate behaviors of a heterogeneous network through simulations and present a rigorous framework to help understand its equilibrium efficiency and fairness properties. By identifying an optimization problem associated with *every* equilibrium, we show that every equilibrium is Pareto efficient and provide an upper bound on efficiency loss due to pricing heterogeneity. On fairness, we show that intra-protocol fairness is still decided by a utility maximization problem while inter-protocol fairness is the part over which we don't have control. However it is shown that we can achieve any desirable inter-protocol fairness by properly choosing protocol parameters. Second, we propose a simple slow timescale source-based algorithm to decouple bandwidth allocation from router parameters and flow arrival patterns and prove its feasibility. The scheme needs only local information.**

I. Introduction

Congestion control algorithm in TCP (Transmission Control Protocol), first introduced in [8], has made important contributions for enabling the explosive growth of the Internet. The currently deployed implementation, referred to as TCP Reno in this paper, uses packet loss as congestion signal to dynamically adapt its transmission rate, or equivalently, its window size.[1] It has worked remarkably well in the past, but its limitations in wireless networks and in networks with large bandwidth-delay product have motivated various proposals that use different congestion signals. For example, in addition to loss based protocols such as HighSpeed TCP [7], STCP [15] and BIC TCP [30], schemes that use queueing delay include the early proposals CARD [10], DUAL [27] and Vegas [2], and the recent proposal FAST [11], [28]. Schemes that use one-bit congestion signal include ECN [20], and those that use multi-bit feedback include XCP [13], MaxNet [29], and RCP [4]. Indeed, the Linux operating system already allows users to choose from a variety of congestion control algorithms since kernel version 2.6.13. Clearly, going forward, our network will become more heterogeneous in which protocols that react to *different* congestion signals interact. Yet, our understanding of such a network is rudimentary at best.

[1]All our experiments and simulations use NewReno with SACK. These are enhanced versions of the original Tahoe and Reno, but we will refer them generically as TCP Reno.

More specifically, in a homogeneous network, even though the sources may control their rates using different algorithms, they all adapt to the *same* type of congestion signals, e.g., all react to loss probability, as in the various variants of Reno and TFRC [6], or all to queueing delay, as in Vegas and FAST. For homogeneous networks, there is already a theory, based on network utility maximization, e.g. [14], [16], [17], [18], [19], [31], that can help understand and design network behaviors. In particular, we know that a homogeneous network of general topology always has a unique equilibrium (operating) point and it is Pareto efficient and the fairness associated with it can be well predicted and controlled. More importantly, the allocation depends only on the congestion control algorithms (equivalently, its underlying utility functions) but not on network parameters (e.g., buffer size) or flow arrival patterns, and hence can be designed through the choice of TCP algorithms.

A heterogeneous network (e.g., one shared by TCP Reno and FAST), however, may have multiple equilibrium points, and they cannot all be stable unless the equilibrium is globally unique [23], [24]. Moreover, the bandwidth allocation among heterogenous flows is now coupled with both network parameters and flow arrival patterns! This is illustrated in Section II through simulations that involve Reno and FAST. It implies that in general we cannot predict, nor control, the bandwidth allocation through the current design of congestion control algorithms for heterogeneous networks. In Section III, a simple source-based algorithm is presented that decouples bandwidth allocation from network parameters and flow arrival patterns in a heterogenous network. Moreover, it drives the network to a unique equilibrium which is efficient and fair.

The rest of the paper then provides a rigorous framework that explains the behavior of heterogeneous networks, and extends the algorithm in section III to the general case. We set up the basic model in section IV. By identifying an optimization problem associated with any given equilibrium point, we discuss efficiency in section V-A. Study of fairness then follows in section V-B. Finally, we propose a general scheme to steer an arbitrary heterogeneous network to the unique equilibrium which solves the standard utility maximization problem by updating a linear scaler in the sources' algorithms in a slow timescale (Section VI). The scheme fully explains the solution we proposed in section III and is readily deployable: it needs only local end-to-end information and linear updates. More realistic experiments that are conducted using WAN in Lab are reported to show the algorithm's effectiveness and some of its byproducts (section VII). We conclude in section VIII and provide some possible future extending directions.

II. Two Examples

In this section, we describe two examples to illustrate some bandwidth allocation problems in heterogenous networks. In the next section, we describe a simple algorithm that solves these problems. All simulations use TCP Reno, which uses packet loss as congestion signal, and FAST TCP, which uses queueing delay as congestion signal. Indeed, networks with both Reno and FAST provide an excellent arena to study problems on heterogeneous networks. Both the problems and the solution will be extended to general networks and protocols beyond Reno and FAST in the following sections IV-VI.

The first experiment (Example 1a) shows that when a Reno flow shares a single bottleneck link with a FAST flow, the relative bandwidth allocation depends critically on the link parameter (buffer size): the Reno flow achieves much higher bandwidth than FAST when the buffer size is large and much smaller bandwidth when it is small. This implies that one cannot control the fairness between Reno and FAST through just the design of end-to-end congestion control algorithms, since fairness is now linked to the network parameters, unlike the case of homogeneous networks.

The second experiment (Example 2a) shows that even on a (multi-link) network with fixed parameters, one cannot control the fairness between Reno and FAST because the relative allocation can change dramatically depending on which flow starts first!

A. Example 1a: dependence of bandwidth allocation on network buffer size

FAST [28] is a high speed TCP variant that uses delay as its main control signal. Every 20ms, a FAST flow adjusts its congestion window W according to

$$W \leftarrow \frac{baseRTT}{RTT} W + \alpha \tag{1}$$

In equilibrium, each FAST flow i achieves a throughput $x_i^* = \frac{\alpha}{q_i^*}$, where q_i^* is the equilibrium queueing delay observed by flow i. Hence, α is the number of packets that each FAST flow maintains in the bottleneck links along its path.

In this example, one FAST flow and one Reno flow share a single bottleneck link with capacity of 8.3 pkts per ms (equivalent to 100Mbps with typical packet size) and round trip propagation delay 50ms. The topology is shown in Figure 1. The FAST flow fixes its α parameter at 50 packets.

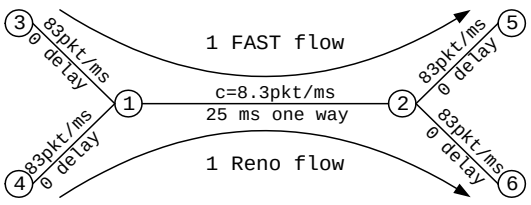
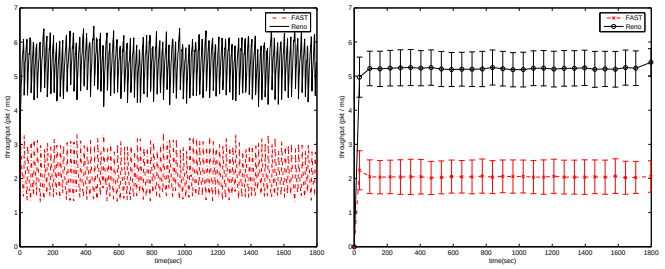


Fig. 1. Single link example.

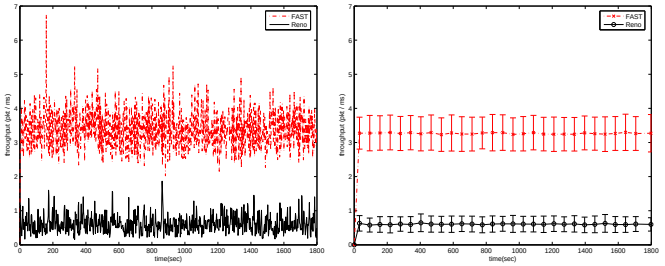
In all of the ns-2 simulations in this paper, heavy-tail noise traffic is introduced in each link with an average rate of 10% of the link capacity.[2] Figure 2 shows the result with a bottleneck buffer size $B = 400$ packets. In this case, FAST gets an average of 2.1 pkts per ms while Reno gets 5.4 pkts per ms. Figure 3 shows the result with $B = 80$ packets. Since the bottleneck buffer size is smaller, the average queue is also smaller. Therefore FAST gets a higher throughput of 3.4 pkts per ms and Reno gets a much lower throughput of 0.6 pkt per ms. In this case, the loss rate is fairly high and the aggregate throughput is much lower (53.6 percent utilization) than the bottleneck capacity due to many timeout events.

In summary, bandwidth sharing between Reno and FAST depends on network parameters in a heterogeneous network, contrary to the case of homogeneous network. This is undesirable since the bandwidth allocation among all competing flows in a network should depend only on their valuation of bandwidth (utility functions) but not on network parameters. In the next section, we propose a simple source-based solution to achieve this.



(a) A sample trajectory (b) Average behavior

Fig. 2. FAST vs. Reno with a buffer size of 400 pkts.



(a) A sample trajectory (b) Average behavior

Fig. 3. FAST vs. Reno with a buffer size of 80 pkts.

B. Example 2a: dependence of bandwidth allocation on flow arrival pattern

The topology of this example is shown in Figure 4. We use RED algorithm [5] and packet marking instead of dropping. The marking probability $p(b)$ of RED is a function of queue length b:

$$p(b) = \begin{cases} 0 & b \le \underline{b} \\ \frac{1}{K}\frac{b-\underline{b}}{\overline{b}-\underline{b}} & \underline{b} \le b \le \overline{b} \\ \frac{1}{K} & b \ge \overline{b} \end{cases} \tag{2}$$

[2] We always present one sample figure on the left and the summary figure on the right. The sample figure shows the rate trajectory in one simulation run. The rate value is measured every 2 seconds. The summary figure presents the rate trajectory averaged over 20 simulation runs with different random seeds. Each point in the summary figure represents the average throughput over a period of one minute. The error bars are also shown in the figure.

where $\underline{b}$, $\overline{b}$ and K are RED parameters. Links 1-2 and 3-4 are both configured with 9.1pkts per ms capacity (equivalent to 111 Mbps), 30 ms one-way propagation delay, and a buffer of 1500 packets. Their RED parameters are ($\underline{b}$, $\overline{b}$, K) = (300, 1500, 10000). Link 2-3 has a capacity of 13.8 pkts per ms (166 Mbps) with 30 ms one-way propagation delay and a buffer size of 1500 packets. Its RED parameters are set to (0, 1500, 10).

There are eight Reno flows on path 1-2-3-4, utilizing all three links, with one-way propagation delay of 90 ms. There are two FAST flows on each of paths 1-2-3 and 2-3-4. Both of them have one-way propagation delay of 60 ms. All FAST flows use a common $\alpha = 50$ packets.

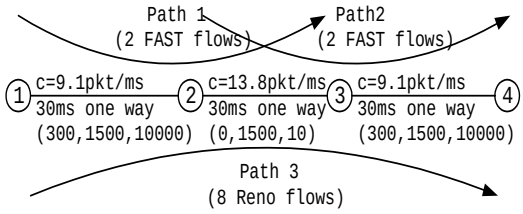
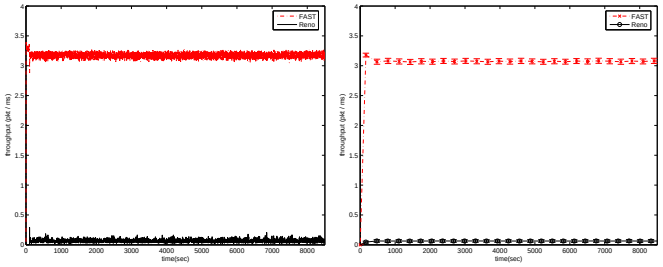


Fig. 4. Multiple equilibria scenario.

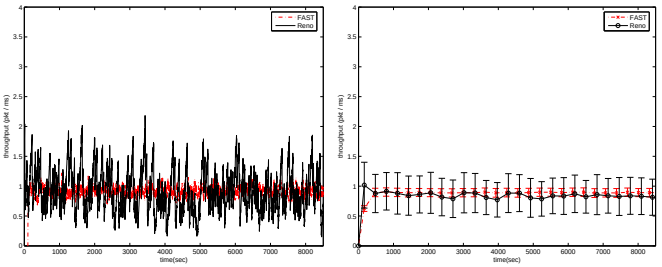
Two sets of simulations have been carried out with different starting times for Reno and FAST flows. The intuition is that if FAST flows start first, link 2-3 will be saturated and links 1-2 and 3-4 will not. Since the RED dropping slope of link 2-3 is steep, when Reno flows join, they will experience so many losses that links 1-2 and 3-4 will remain unsaturated. If Reno flows start first, on the other hand, links 1-2 and 3-4 are saturated while link 2-3 is not because link 2-3 has a higher capacity. Since the RED dropping slopes of link 1-2 and 3-4 are not steep, they can generate enough queueing delay to squeeze FAST flows when they join and keep link 2-3 unsaturated. In the simulations, one set of flows (Reno or FAST) starts at time zero, and the other set of flows starts at the 100th second. We present the throughput achieved by one of the FAST flows and one of the Reno flows (Figures 5 and 6). Each point in the summary figures represents the average rate over 5 minutes.



(a) A sample trajectory (b) Average behavior

Fig. 5. Bandwidth shares of Reno and FAST when FAST starts first.

Since the difference of rate allocations in these two figures is far greater than the standard deviation, it is clear that the network has reached very different equilibria depending on which flows start first. In short, bandwidth sharing in heterogeneous networks depends on which type of TCP starts first and becomes not predictable. This is also undesirable.



(a) A sample trajectory (b) Average behavior

Fig. 6. Bandwidth shares of Reno and FAST when Reno starts first.

Algorithm 1 α adaptation algorithm

1) Every α update interval (2 minutes by default), calculate:

$$\alpha^* = \frac{q}{lw}$$

q and l are average queueing delay and average packet loss rate over the interval, w is a parameter. Then

$$\alpha = \begin{cases} \min\{(1+\delta)\alpha, \alpha^*\} & \text{if } \alpha < \alpha^* \\ \max\{(1-\delta)\alpha, \alpha^*\} & \text{if } \alpha > \alpha^* \end{cases}$$

δ is 0.1 by default.

2) Every window update interval (20ms by default), run FAST algorithm (1).

III. One Solution

We now propose a simple source-based algorithm (Algorithm 1) for FAST flows to solve the problems on unfairness and unpredictable parameter sensitivity illustrated by the examples in the previous section. Complete development of the algorithm, together with the theoretical foundation and simulation verifications of the solution for general cases, which are beyond networks with just Reno and FAST, form the rest of the paper after this section. The solution tries to achieve a unique equilibrium that efficiently and fairly utilizes the bandwidth. The allocation is independent of router setting. The solution only uses end-to-end local information that is available to each flow and only requires simple parameter updates, such as the linear parameter α in FAST.

This α adaptation algorithm, Algorithm 1, fine-tunes the value of α according to the signals of queue delay and loss in a large time scale (several RTTs). The basic idea of the solution is that FAST should adjust its aggressiveness (α) to the proper level by looking at the ratio of end-to-end queueing delay and end-to-end loss. In other words, FAST also reacts to loss in a slow timescale.

A. Example 1b: independence of bandwidth allocation on buffer size

We repeat the simulations in Example 1a with Algorithm 1, w is set to be 125s[3]. Figure 7, Figure 8, Figure 9 and Figure 10 should be compared with Figure 2, Figure 3, Figure 5 and Figure 6 correspondingly.

With Algorithm 1, FAST achieves 3.4 pkts per ms with buffer size of 400 and 3.2 pkts per ms with buffer size of

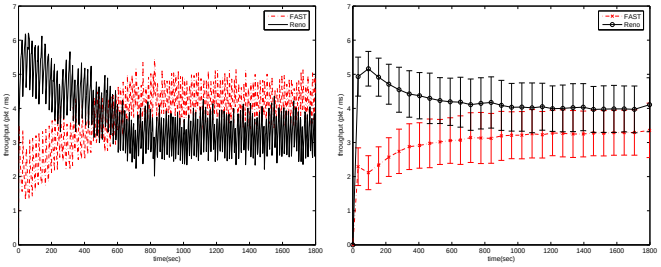
[3] w determines the equilibrium bandwidth share. Formally, it is stated in (23)-(24). Here w is chosen so that Reno and FAST get equal rates.

80, while Reno gets 4.2 pkts per ms and 4.1 pkts per ms, respectively. The fairness is greatly improved and essentially independent of buffer size now, which we summarize in table I by listing the ratio of bandwidth that Reno gets to what FAST gets in different scenarios. We also note that the utilization of the link for $B = 80$ case increases dramatically from 53.6 percent to 97.7 percent. This point will be further discussed in Experiment 3 in section VII.

	B=400	B=80
Without Algorithm 1	5.4/2.1=2.6	0.6/3.4=0.18
With Algorithm 1	4.2/3.1=1.4	4.1/3.2=1.3

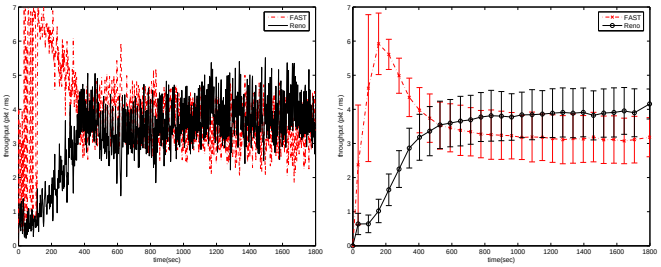
TABLE I

RATIO OF RENO'S RATE AND FAST'S RATE



(a) A Sample (b) Average Behavior

Fig. 7. FAST vs. Reno, with buffer size of 400 pkts.

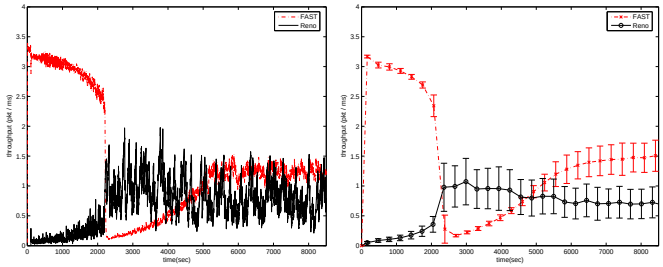


(a) A Sample (b) Average Behavior

Fig. 8. FAST vs. Reno, with buffer size of 80 pkts.

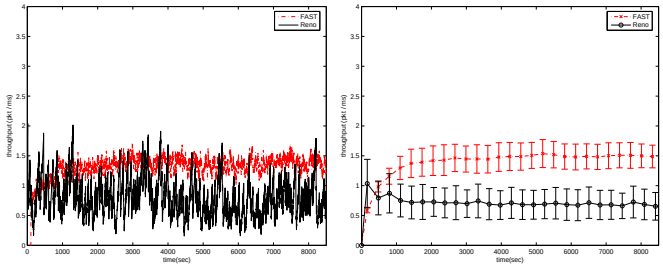
B. Example 2b: independence of bandwidth allocation on flow arrival pattern

We repeat the simulations in Example 2a with Algorithm 1, w is set to be 1820s. Figure 9 and Figure 10 show the effect of α adaptation in the multiple-bottleneck case that we introduced in Example 2. As we will prove in Theorem 6, there is always a unique equilibrium if we adapt α according to Algorithm 1. In this particular case, this single equilibrium is around the point where each Reno flow gets a throughput of 0.6 pkts per ms and each FAST flow gets 1.5 pkts per ms. At this single equilibrium, link 1 and link 3 are the bottleneck links. In Figure 9, FAST flows start on time zero and link 2 becomes the bottleneck. When Reno flows join on the 100th second, the ratio of queue delay to loss at link 2 is much higher than the target value. The FAST flows hence reduce their α values gradually and the set of bottleneck links switches from link 2 to link 1 and 3 around the 2000th second. After that, FAST flows and Reno flows converge to the unique equilibrium.



(a) A Sample (b) Average Behavior

Fig. 9. FAST starts first.



(a) A Sample (b) Average Behavior

Fig. 10. Reno starts first.

So far, we have used TCP Reno and FAST to show the problems of networks using both loss and delay as congestion measures. A solution is also proposed and validated by simulations. The remainder of the paper will be devoted to studying this topic in a general setting beyond networks with just Reno and FAST, and showing the intuition and correctness of this solution, which also involves developing understanding of important properties of heterogeneous congestion control, including efficiency and fairness.

IV. MODEL

A. Notations and Assumptions

Consider a network consisting of a set of L links, indexed by $l = 1, \ldots, L$, with fixed finite capacities c_l. We sometimes abuse notation and use L to denote both the number of links and the set $L = \{1, \ldots, L\}$ of links. Each link has a price p_l as its congestion measure. There are J different congestion control protocols indexed by superscript j, and N^j sources using protocol j, indexed by (j, i) where $j = 1, \ldots, J$ and $i = 1, \ldots, N^j$. The set of links used by source (j, i) is denoted by $L(j, i)$, and the total number of sources by $N := \sum_j N^j$.

The $L \times N^j$ routing matrix R^j for type j sources is defined by $R^j_{li} = 1$ if source (j, i) uses link l, and 0 otherwise. The overall routing matrix is denoted by

$$R = \left[\begin{array}{cccc} R^1 & R^2 & \cdots & R^J \end{array} \right]$$

Even though different classes of sources react to different prices, e.g. Reno to packet loss probability and Vegas/FAST to queueing delay, the prices are related. We model this relationship through a price mapping function that maps a

common "intrinsic" price (e.g. queue length) at a link to different prices (e.g. loss probability and queueing delay) observed by different sources. Formally, every link l has a price p_l. A type j source reacts to the "effective price" $m_l^j(p_l)$ in its path, where m_l^j is a price mapping function, which can depend on both the link and the protocol type. The exact form of m_l^j depends on the AQM algorithm used at the link; see (2) for links with RED.[4] Let $m^j(p) = (m_l^j(p_l), l = 1, \dots L)$ and $m(p) = (m^j(p_l), j = 1, \dots J)$. The aggregate prices for source (j, i) is defined as

$$q_i^j \;=\; \sum_l R_{li}^j m_l^j(p_l) \tag{3}$$

Let $q^j = (q_i^j, i = 1, \dots, N^j)$ and $q = (q^j, j = 1 \dots, J)$ be vectors of aggregate prices. Then $q^j = \left(R^j\right)^T m^j(p)$ and $q = R^T m(p)$.

Let x^j be a vector with the rate x_i^j of source (j, i) as its ith entry, and x be the vector of x^j

$$x \;=\; \left[\ (x^1)^T, (x^2)^T, \dots, (x^J)^T\ \right]^T$$

Source (j, i) has a utility function[5] $U_i^j(x_i^j)$ that is strictly concave increasing in its rate x_i^j. Let $U = (U_i^j, i = 1, \dots, N^j, j = 1, \dots, J)$.

With the above notation, we refer to (c, m, R, U) as a *network*, where (in general) z denotes the (column) vector $z = (z_k,\ \forall k)$. The following basic assumptions are adopted, as in [23], [25] that studied the existence and uniqueness of equilibrium for heterogeneous protocols.

A1: Utility functions U_i^j are strictly concave increasing, and twice continuously differentiable in their domains. Price mapping functions m_l^j are continuously differentiable and strictly increasing with $m_l^j(0) = 0$.

A2: For any $\epsilon > 0$, there exists a number $p_{\max}$ such that if $p_l > p_{\max}$ for link l, then

$$x_i^j(p) < \epsilon \text{ for all } (j, i) \text{ with } R_{li}^j = 1$$

B. Network Model

We consider the "dual algorithm" [17] where sources select transmission rates that maximize their utility minus bandwidth cost, and network links adjust bandwidth prices according to the utilization of the links:

$$x_i^j\left(q_i^j\right) \;=\; \left[\left(U_i^j\right)'^{-1}\left(q_i^j\right)\right]^+$$

$$\dot{p}_l(t) \;=\; y_l(p(t)) - c_l =: f_l(p(t)) \tag{4}$$

Under the assumptions in this paper, $\left(U_i^j\right)'^{-1}\left(q_i^j\right) > 0$ for all the prices p that we consider, and hence we can ignore the projection $[\cdot]^+$ and assume, without loss of generality, that

$$x_i^j\left(q_i^j\right) \;=\; \left(U_i^j\right)'^{-1}\left(q_i^j\right) \tag{5}$$

(5) is nothing but the responsive function of TCP which determines source average sending rate based on its observed end-to-end congestion signal.

As usual, we use $x^j\left(q^j\right) = \left(x_i^j\left(q_i^j\right), i = 1, \dots, N^j\right)$ and $x(q) = \left(x^j\left(q^j\right), j = 1, \dots, J\right)$ to denote the vector-valued functions composed of x_i^j. Since $q = R^T m(p)$, we often abuse notation and write $x_i^j(p), x^j(p), x(p)$. Define the aggregate source rates $y(p) = (y_l(p), l = 1, \dots, L)$ at links l as:

$$y^j(p) \;=\; R^j x^j(p), \qquad y(p) \;=\; Rx(p) \tag{6}$$

In equilibrium, the aggregate rate at each link is no more than the link capacity, and they are equal if the link price is strictly positive. Formally, we call p an *equilibrium price* (or a *network equilibrium* or an *equilibrium*) if it satisfies (from (3), (5), (6))

$$P(y(p) - c) = 0, \quad y(p) \le c, \quad p \ge 0 \tag{7}$$

where $P := \text{diag}(p_l)$ is a diagonal matrix.

When all sources react to the same price, then the equilibrium described by (3), (5)-(7) is the unique solution of the following utility maximization problem defined in [14] and its Lagrange dual [17]:

$$\max_{x \ge 0} \qquad \sum_i U_i(x_i) \tag{8}$$

$$\text{subject to} \qquad Rx \;\le\; c \tag{9}$$

where we have omitted the superscript $j = 1$. The strict concavity of U_i guarantees the existence and uniqueness of the optimal solution of (8)–(9).

For heterogeneous case, the utility maximization problem no long underlies the equilibrium described by (3), (5)-(7). The current theory cannot be directly applied and great difficulties arise when exploring even the basic questions such as uniqueness of equilibrium [23], [25].

V. Analysis of Efficiency and Fairness

A. Efficiency

In this subsection, efficiency of equilibrium of networks with heterogeneous protocols is explored. We first make the following key observation, which not only leads to the remaining results of this subsection, but also is the starting point of our algorithm design in section VI.

Theorem 1. *Given an equilibrium p^*, there exists a positive vector γ, such that the equilibrium rate vector $x^*(p)$ is the unique solution of following problem:*

$$\max_{x \ge 0} \qquad \sum_{i,j} \gamma_i^j U_i^j(x_i^j) \tag{10}$$

$$\textit{subject to} \qquad Rx \le\; c \tag{11}$$

Proof. The KKT (Karush-Kuhn-Tucker) optimality conditions for (10), (11) are:

$$\gamma_i^j\left(U_i^j\right)'(x_i^j) \;=\; \sum_l R_{il}^j p_l \quad \text{for all } (i, j) \tag{12}$$

$$p^T(Rx - c) \;=\; 0 \tag{13}$$

$$Rx - c \;\le\; 0 \tag{14}$$

[4] One can also take the price p_l^j used by one of the protocols, e.g. queueing delay, as the common price p_l. In this case the corresponding price mapping function is the identity function, $m_l^j(p_l) = p_l$.

[5] Almost all TCP variants people have proposed or deployed can be shown to implicitly maximize some strictly concave increasing utility functions [18]. One can also start from a given utility function (e.g. from application layer) and derive protocols. Here we take this view and use utility function to represent the exact form of congestion protocol.

where the (x, p) are the primal-dual variables. We now claim these conditions are satisfied with equilibrium rates and prices (x^*, p^*) by choosing

$$\gamma_i^j = \frac{\sum_l R_{il}^j p_l^*}{\sum_l R_{il}^j m_l^j(p_l^*)} \tag{15}$$

To see this, note (13) and (14) are conditions for equilibrium. After substituting (15) into (12), we have

$$\left(U_i^j\right)'(x_i^{j*}) = \sum_l R_{il}^j m_l^j(p_l^*) \tag{16}$$

That is consistent with equations (3) and (5) that are used to define equilibrium. □

Theorem 1 gives an underlying convex optimization problem that an equilibrium solves, which is amenable to distributed solutions as we will see in section VI. However, it is important to note that this optimization problem itself depends on equilibrium. Hence it cannot be used to find equilibrium directly, nor does it guarantee existence and uniqueness as in the single-protocol case [23], [25].

As stated by the celebrated first fundamental theorem of welfare economics, any competitive equilibrium is Pareto efficient. That explains the most basic reason that congestion signals are used to regulate source rates and hence realize bandwidth allocation. We know the unique equilibrium is Pareto efficient when there is a single price. Now we can also show that the same holds for networks with heterogeneous protocols as a direct corollary of Theorem 1:

Corollary 2. *All equilibrium points are Pareto efficient.*

Pareto efficiency can be viewed as a qualitative requirement for an efficient allocation. However, it does not give a quantitative description. Instead, aggregate utility (social welfare) is the standard criterion for efficiency. As shown in (8)-(9), for homogeneous cases, the unique equilibrium achieves the maximum aggregate utility. For heterogeneous protocol cases, we now study efficiency loss by lower-bounding the ratio of the achieved aggregate utility to its maximum.

Theorem 3. *Assume all utility functions are nonnegative, i.e., $U(x) \geq 0$. Suppose the optimal aggregate utility is U^* and $\hat{U}$ is the achieved aggregate utility at an equilibrium ($\hat{x}$) of a network with heterogeneous protocols. Then*

$$\frac{\hat{U}}{U^*} \geq \frac{\underline{\gamma}}{\overline{\gamma}} \tag{17}$$

where $\underline{\gamma}$ and $\overline{\gamma}$ are the lower and upper bounds of γ_i^j[6], i.e., $\underline{\gamma} \leq \gamma_i^j \leq \overline{\gamma}$.

Proof. Assume $\hat{x}$ is one of the solutions of Theorem 1, then

$$\max_x \sum_{i,j} \gamma_i^j U_i^j(x_i^j) = \sum_{i,j} \gamma_i^j U_i^j(\hat{x}_i^j) \leq \overline{\gamma}\hat{U} \tag{18}$$

On the other hand,

$$\max_x \sum_{i,j} \gamma_i^j U_i^j(x_i^j) \geq \underline{\gamma} \max_x \sum_{i,j} U_i^j(x_i^j) = \underline{\gamma} U^* \tag{19}$$

[6]Both $\underline{\gamma}$ and $\overline{\gamma}$ can be bounded using $\dot{m}_l^j$. For example, for a network with both loss based and delay based protocols and assuming RED is used, the slopes of RED at different links can provide information on $\dot{m}_l^j$ and therefore $\underline{\gamma}$ and $\overline{\gamma}$.

Combining the two equalities above, we get

$$\frac{\hat{U}}{U^*} \geq \frac{\underline{\gamma}}{\overline{\gamma}}$$

□

It has been well known for long time that price can serve as the "invisible hand" to coordinate competing users and realize optimal resource allocation. That however requires two basic assumptions. The first assumption is that users are all price takers. If instead they are noncooperative game players, there will be efficiency loss. Such "price of anarchy" was recently bounded from above for both routing [21] and congestion control [12]. The second assumption is the homogeneity of price that all users see, which does not hold in networks with more than one type of congestion control protocols. Our result above quantifies the "price of heterogeneity".

B. Fairness

In this subsection, we study fairness in networks shared by heterogeneous congestion control protocols. Two questions we address are: how the flows within each protocol share among themselves (intra-protocol fairness) and how these protocols share bandwidth in equilibrium (inter-protocol fairness). The results here generalize corresponding theorems in [24].

1) Intra-protocol fairness: As indicated by (8)–(9), when the network is shared only by flows using the same congestion signal, the equilibrium flow rates are the unique optimal solution of the utility maximization problem. In other words, the utility functions describe how the flows share bandwidth among themselves. When flows using different congestion signals share the same network, it turns out that this feature is still preserved "locally" within each protocol, as we now show.

Theorem 4. *Given an equilibrium $(\hat{x}, \hat{p}) \geq 0$, let $\hat{c}^j := R^j \hat{x}^j$ be the total bandwidth consumed by flows using protocol j at each link. The corresponding flow rates $\hat{x}^j$ are the unique solution of:*

$$\max_{x^j \geq 0} \quad \sum_{i=1}^{N^f} U_i^j(x_i^j) \qquad \textit{subject to} \quad R^j x^j \leq \hat{c}^j \tag{20}$$

Proof: Since $(\hat{x}^j, \hat{p}^j) \geq 0$ is an equilibrium, from (3) to (7), we have

$$\left(U_i^j\right)'\left(\hat{x}_i^j\right) \quad = \quad \sum_l R_{li}^j \hat{p}_l^j \quad \text{for} \quad i = 1, ..., N^f$$

This, together with (from the definition of $\hat{c}^j$)

$$\sum_i R_{li}^j \hat{x}_i^j \leq \hat{c}_l^j, \qquad \hat{p}_l^j \left(\sum_i R_{li}^j \hat{x}_i^j - \hat{c}_l^j\right) = 0, \qquad \forall l$$

forms the necessary and sufficient condition for $\hat{x}^j$ and $\hat{p}^j$ to be optimal for (20) and its dual respectively. □

Note that in Theorem 4, the "effective capacities" $\hat{c}^j$'s are not preassigned. They are the outcome of competition among flows using different congestion prices and are related to inter-protocol fairness, which we now discuss.

2) Inter-protocol fairness: Even though flows using different congestion signals individually solve a utility maximization problem to determine their intra-protocol fairness, they in general do not jointly solve any predefined convex utility maximization problem. This makes the study of inter-protocol fairness hard. Here we provide a feasibility result, which says any reasonable inter-protocol fairness is achievable by linearly scaling congestion control algorithms.

Assume flow (j,i) has a parameter μ_i^j with which it chooses its rate in the following way:

$$x_i^j\left(q_i^j\right) = \left(U_i^j\right)'^{-1}\left(\frac{1}{\mu_i^j}q_i^j\right) \tag{21}$$

For example, if we consider FAST's utility function $\alpha\log(x)$, then the α parameter in the protocol can be viewed as μ here. Our main result in this subsection says for a network with J protocols, if $J-1$ protocols have their linear scaler vectors μ^j, then there exists a μ vector such that one of the resulting equilibria with that μ can achieve any predefined bandwidth partition. Before we get to the theorem itself, we first characterize the feasible set of predefined bandwidth allocation.

Assume that except for $j=J$, flow (i,j) has parameter μ_i^j. Or equivalently, we can define $\mu_i^J=1$. The equilibrium rates x^j clearly depend on parameter μ. For $j=1,2,...J-1$, let $\overline{x}^j(\mu)$ be the unique rate vector of flows using protocol j if there were no other protocols in the network. Let $\underline{x}^j(\mu)$ be the unique rates of type j flows if network capacity were $(c-\sum_{k\neq j}R^k\overline{x}^k)^+$.

Let

$$X \;:=\; \{\; x|\,\underline{x}^j(\mu)\le x^j\le\overline{x}^j(\mu),\;\mu\ge 0, Rx\le c\}$$

X includes all possible rates of flows using protocol j if they were given strict priority over other flows or if others were given strict priority over them, and all rates in between. In this sense X contains the entire spectrum of inter-protocol fairness among different protocols. The next result says that every point in this spectrum is achievable by an appropriate choice of parameter μ.

Let $S(\mu)$ denote the set of equilibrium rates of flows when the protocol parameter is μ. Clearly, equilibrium is characterized by (3), (21), (6) and (7).

Theorem 5. *For every link l, assume there is at least one type J flow that only uses that link. Given any $x\in X$, there exists an $\mu\ge 0$ such that $x\in S(\mu)$.*

Proof: Given any $x\in X$, the capacity for all type J flows is $c-\sum_{k\neq J}R^k(x^k)$. Since $Rx\le c$ (for all coordinates), we have $c-\sum_{k\neq J}R^k(x^k)\ge(x^J)$, which is greater than or equal to 0. Hence the following utility maximization problem solved by flows of type J is feasible:

$$\begin{aligned}\max_{x^J\ge 0}\quad & \sum_i U_i^J(x_i^J)\\ \text{subject to}\quad & R^Jx^J\;\le\; c-\sum_{k\neq J}R^k(x^k)\end{aligned}$$

Let p^J be the associated Lagrange multiplier vector. By the assumption that every link has at least one single-link type J flow, we know $p_l^J>0$ for all l. Choose $(\mu_i^j)=\frac{\sum_l R_{li}^j m_l^j((m^J)_l^{-1}(p_l^J))}{(U_i^j)'(x_i^j)}$. It can be checked that all equations that characterize equilibrium ((3), (21), (6) and (7)) are satisfied. □

In general, one can view Theorem 1 as defining fairness of flows using heterogeneous protocols and can conclude that price mapping functions (router parameters) affect fairness (supported by Example 1a). Clearly, if one can choose price mapping functions, one can achieve any predefined fairness. More interestingly, Theorem 5 implies that given any reasonable fairness among flows using different congestion signals, in terms of a desirable rate allocation x, there exists a protocol parameter vector μ that can achieve it without changing parameters inside the network. It is however yet unclear how to compute μ dynamically in practice using only local information. In section VI, we will discuss distributed algorithms to compute a particular μ, which will result in the optimal bandwidth allocation.

VI. Slow timescale update

A. Theoretical Foundation

As pointed out in Corollary 2, all equilibria are Pareto efficient. However, based on analysis in section V, large efficiency loss may occur and no guarantee on fairness can be provided. These motivate us now to turn from analysis to design, and develop a readily implementable control mechanism that "drives" any network with heterogeneous congestion control protocols to any desired operating point with a fair and efficient bandwidth allocation. It generalizes the Algorithm 1 in section III and explains the intuition and theoretical foundation of it. The central problems that motivate our study here include: What is the equilibrium the system should be driven to? Can we make it unique? Will it solve any global optimization problem? How to do that in a distributed way? In this section, we propose an answer by introducing slow timescale updating. Our target equilibrium is still the maximizer of some weighted aggregate utility. The first step is to set up the existence and uniqueness of such a solution.

Theorem 6. *For any given network (c,m,U,R), for any positive vector w, there exists a unique positive vector μ such that, if every source scales their own prices by μ_i^j, i.e.,*

$$x_i^j=\left(U_i^j\right)'^{-1}\left(\frac{1}{\mu_i^j}\sum m_l^j(p_l)\right) \tag{22}$$

then, at equilibrium(x,p), x solves

$$\max_{x\ge 0}\quad \sum_{(i,j)}\frac{1}{w_i^j}U_i^j(x_i^j) \tag{23}$$

$$\textit{subject to}\quad Rx\;\le\;c \tag{24}$$

Moreover,

$$\mu_i^j=\frac{1}{w_i^j}\frac{\sum_{l\in L(j,i)}m_l^j(p_l)}{\sum_{l\in L(j,i)}p_l}$$

Proof. We claim that the optimality conditions of (23) and (24) are the same as equations that characterize the equilibrium of

Algorithm 2 Two timescale control scheme

1) Every source chooses its rate by $x_i^j(t) = (U')^{-1}(\frac{q_i^j(t)}{\mu_i^j(t)})$;
2) Every source updates its μ_i^j by $\mu_i^j(t+T) = \mu_i^j(t) + \kappa_i^j \left(\frac{\sum_{l \in L(j,i)} m_l^j(p_l(t+T))}{\sum_{l \in L(j,i)} p_l(t+T)} - \mu_i^j(t) \right)$ where κ_i^j is the stepsize for flow (j,i) and T is large enough so that the fast timescale dynamics among x and p can reach steady state.

the above system ((3), (22), (6) and (7)). Capacity constraints, nonnegativity, and complementary slackness are obviously the same. We only need to check the relation between rates and prices at equilibrium. Those are

$$\mu_i^j \left(U_i^j\right)'(x_i^j) = \sum_{l \in L(j,i)} m_l^j(p_l) \tag{25}$$

and

$$\mu_i^j = \frac{1}{w_i^j} \frac{\sum_{l \in L(j,i)} m_l^j(p_l)}{\sum_{l \in L(j,i)} p_l} \tag{26}$$

Combining them, we get

$$\frac{1}{w_i^j} \left(U_i^j\right)'(x_i^j) = \sum_{l \in L(j,i)} p_l \tag{27}$$

which is the relation between x and p specified by the optimality conditions of problem (23)-(24). On the other hand, given x and p that satisfy (27), one can always define μ by (26), and (25) will also be satisfied. □

Parameter w enables us to measure fairness and to achieve any desired fair bandwidth allocation. But we need all sources to have access to *one* common price. Moreover, Theorem 6 suggests Algorithm 2 as a two-timescale scheme to control the operation point of networks with heterogenous congestion control protocols. The essential idea in Algorithm 2 is that by reacting to the same price in slow timescale, uniqueness and fairness of equilibrium is guaranteed in the long run. Yet the algorithm allows sources to react to their own effective prices $m_i^j(p_l(t))$ in fast timescale. This flexibility in timescales is important in practice when, for example, the link prices p_l are loss probability that are hard to reliably estimate at the fast timescale. The slow timescale algorithm only updates a linear scaler, which is readily implementable, e.g., this corresponds to update a parameter α in FAST. Indeed, if we specialize Algorithm 2 to FAST/Reno networks using loss as the common price p, we get Algorithm 1 in section III. In general, as we can always choose $m_l^j(p_l) = p_l$ for a particular j, say $j = 1$. Then we have $\mu_i^1 = 1$, which means sources of type 1 don't need to adapt. This is crucial for deployment as only new protocols need to adapt while the current Reno does not.

B. Numerical Results

Throughout this subsection, we provide some numerical results to further validate the effectiveness of Algorithm 2. For simplicity we choose w to be a vector with all components being 1, i.e., we attempt to maximize the aggregate utility.

Experiment 1: L=3 with multiple equilibria
In this experiment, we use the following example that has multiple equilibria [23]. The network is shown in Figure 11 with three unit-capacity links, $c_l = 1$. There are three different protocols with the corresponding routing matrices

$$R^1 = I, \quad R^2 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}^T, \quad R^3 = (1,1,1)^T$$

The price mapping functions are linear: $m_l^j(p_l) = k_l^j p_l$ where

$$K^1 = I, \quad K^2 = \text{diag}(5,1,5), \quad K^3 = \text{diag}(1,3,1)$$

Utility functions of sources (j,i) are

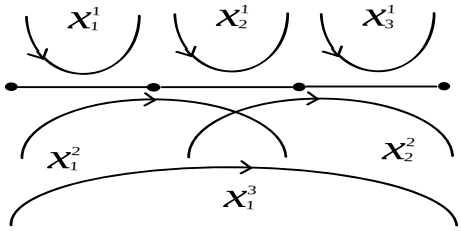
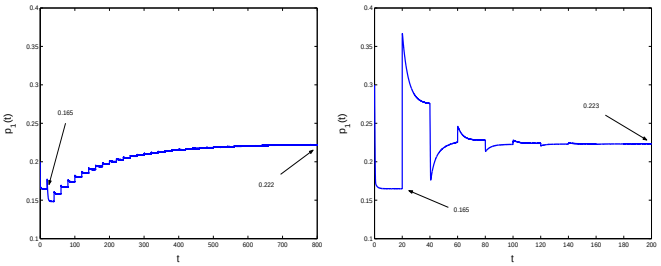


Fig. 11. A three-link network with three equilibria

$$U_i^j(x_i^j, \alpha_i^j) = \begin{cases} \beta_i^j (x_i^j)^{1-\alpha_i^j}/(1-\alpha_i^j) & \text{if } \alpha_i^j \neq 1 \\ \beta_i^j \log x_i^j & \text{if } \alpha_i^j = 1 \end{cases}$$

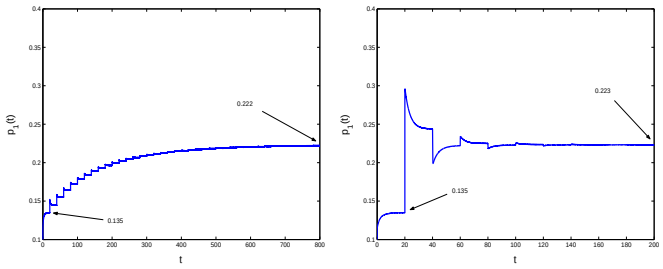
with appropriately chosen positive constants α_i^j and β_i^j [23]. These utility functions can be viewed as a weighted version of the α-fairness utility functions proposed in [19]. μ_i^j's are updated every 20 time units. We show that starting from different initial conditions, although the system reaches different equilibria after the first iteration, it nevertheless finally reaches the unique optimal equilibrium with $p_1^* = 0.222$.
Case 1: We start with initial point $p_1(0) = p_2(0) = p_3(0) = 0.3$. After the first iteration, the network goes to an equilibrium ($p_1 = p_3 = 0.165$, $p_2 = 0.170$). $p_1(t)$ with different updating stepsize κ_i^j is shown in Figures 12.



(a) stepsize $\kappa_i^j = 0.1$ (b) stepsize $\kappa_i^j = 0.5$

Fig. 12. Case 1: $p_1(t)$ with different κ_i^j

Case 2: We choose another initial point $p_1(0) = p_3(0) = 0.1$, $p_2(0) = 0.3$ As shown in Figure.13. After the first iteration, the system reaches another equilibrium, $p_1 = p_3 = 0.135$ and $p_2 = 0.230$. However finally, the system still reaches the same steady state as in Figure 13.



(a) stepsize $\kappa_i^j = 0.1$ (b) stepsize $\kappa_i^j = 0.5$

Fig. 13. Case 2: $p_1(t)$ with different κ_i^j

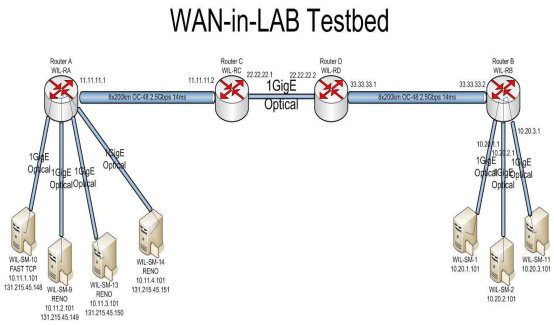
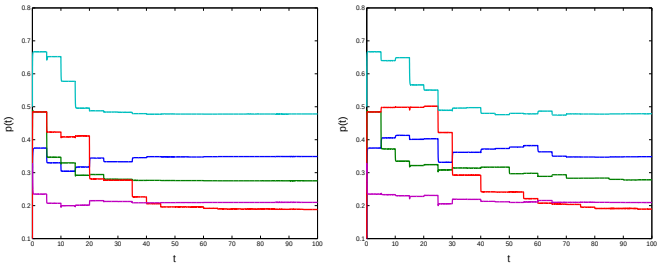


Fig. 15. WAN in Lab experiment setup

Experiment 2: L=5 with asynchronous update

In this experiment, the network has five links and 15 flows. Algorithm 2 is tested in an asynchronous environment. We assume that every five time units, flows can update their μ_i^j and they do so with certain probability. Hence every five time units, only a portion of flows update their μ_i^j. We randomly set link capacities uniformly between 1 to 10, price mapping functions are $m^1(p) = p$ and $m^2(p) = p^\alpha$, where α is randomly chosen between 0.5 to 5 with uniform distribution. Flows 1 to 5 use links 1 to 5 correspondingly while a random routing matrix with entries 0 or 1 with equal probability is used to define routes for other flows. Finally each flow randomly chooses to use price 1 or 2 with equal probability.

All of the 1000 trials converge to the right target point. Some typical convergence patterns are shown in Figure 14 where the five curves correspond to the p value of the five links. It shows clearly that although asynchronism causes longer convergence time, the system still converges to the same equilibrium.



(a) update with probability 0.6 (b) update with probability 0.3

Fig. 14. p(t) with different probability of updating

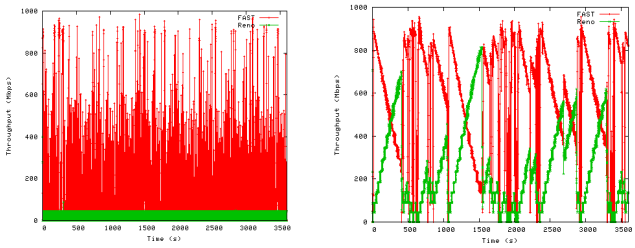
VII. WAN in Lab Experiments

The objective of experiments in this section is to show the effectiveness and some other features of Algorithm 2 (Algorithm 1 when we focus on networks with TCP Reno and FAST) in a more realistic setting. We achieve that by carrying out experiments with TCP Reno and FAST in the hardware testbed of WAN in Lab [26], which is a wide area network consisting of an array of reconfigurable routers, servers and clients, and by considering some previously ignored scenarios (e.g., small buffer size, only FAST flows). We test our algorithm with a single bottleneck link shown in Figure 15.

Experiment 3: small buffer size

As we have seen in Example 1a and Example 1b, Algorithm 1 can dramatically increase link utilization when buffer size B is not too larger than α_0. In this experiment, we go to the extreme region when $B < \alpha_0$. As every FAST flow tries to maintain α packets in the queues along its path. Clearly if $B < \alpha_0$, constant high packet loss rate will occur and both Reno and FAST will have very poor throughput. We show a a byproduct of Algorithm 1 that it can adjust α automatically to a proper value to fit the inside network parameter B.

One FAST and one Reno compete for bandwidth of a bottleneck link of 1Gbps (80pkts/ms) capacity. The buffer capacity B is 480pkts. The initial α is set to be α_0=800pkts. The results are summarized in Figure 16. As the left part of the figure shows, both Reno and FAST get very low throughput due to the high packet loss rate (FAST: 135Mbps; Reno: 22Mbps). However, using Algorithm 1, FAST decreases its α as it sees high loss and finally both flows get high throughput (FAST: 593Mbps; Reno: 246Mbps). The utilization is increased dramatically from 15.7 percent to 83.9 percent.



(a) Without Algorithm 1 (b) With Algorithm 1

Fig. 16. Bandwidth partition between Reno and FAST

Experiment 4: only FAST flows

Although the slow timescale update shows desirable properties in various tests we have discussed so far, there is a problem we have not touched, namely the case when there are only FAST flows in a network. As FAST is designed to achieve a steady state with no loss, flows will keep increasing their α according to Algorithm 1 until the buffer is filled and loss is generated. This is not desirable and we propose to turn off the slow timescale update when a FAST flow has not seen any loss for a certain amount of time (ten seconds by default). We conduct a test using three FAST flows all with α_0=200pkts sharing the same link as in Experiment 3. The throughput trajectories are shown in Figure 17. We can see that after a period of adjusting, all flows are stabilized. The steady state throughputs are 128Mbps, 234Mbps and 566Mbps, which result in a high

utilization of 92.8 percent even though the initial sum of α (600 pkts) exceeds the buffer capacity (480 pkts). However, this introduces potential fairness problem as we cannot control the exact α values when the updating algorithm stops. For example, instead of achieving perfect fairness with a Jain index [9] of 1, we have 0.733 in this experiment. We tend to think that this short term unfairness is not so important as in practice, flows come and go [1], which will give many chances for existing flows to reshuffle and the random short term unfairness can be averaged out to yield long term fairness.

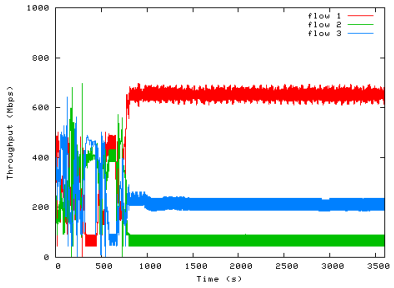


Fig. 17. Bandwidth sharing among FAST flows

VIII. Conclusion

When sources sharing the same network do flow control based on different congestion signals, the existing duality model no longer explains the behaviors of bandwidth allocation. In this paper, we have studied fundamental properties of such networks like efficiency and fairness. In particular, it is shown that qualitatively equilibrium is still Pareto efficient but quantitatively there is efficiency loss about which we can provide an upperbound. On fairness, intra-protocol fairness is still determined by utility maximization problem, while inter-protocol fairness is the part which we don't have control on. However, we can achieve any desirable inter-protocol fairness by properly choosing protocol parameters. Motivated by and based on these results, we propose a distributed scheme to steer the whole network to the unique equilibrium which maximizes aggregate utility. The scheme only needs to update a linear scaler in source algorithm in a slow timescale.

There are a number of ways to extend this work. For example, more efforts are needed to fully clarify the dynamics of the system. Another interesting question would be what is the optimal way for sources to regulate their rates given they have access to multiple congestion signals. Some steps have been taken along this direction by combining delay-based and loss-based congestion control protocols [3], [22].

Acknowledgments: We acknowledge the use of Caltech's WAN in Lab facility funded by NSF (through grant EIA-0303620), Cisco ARTI, ARO (through grant W911NF-04-1-0095), and Corning. We also thank the support from NSF CCF-0448012, CNS-0417607, DARPA HR0011-06-1-0008, and AFOSR FA9550-06-0297.

References

[1] T. Bonald and L. Massoulie. Impact of fairness on Internet performance. In *Proceedings of ACM Sigmetrics*, June 2001

[2] L. Brakmo and L. Peterson. TCP Vegas: end-to-end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–80, October 1995.

[3] C. Casetti, M. Gerla, S. Mascolo, M. Sansadidi and R. Wang. TCP Westwood: end to end congestion control for wired/wireless networks. Wireless Networks Journal, vol.8, pp.467-479, 2002

[4] N. Dukkipati, M. Kobayashi, R. Z. Shen and N .McKeown. Processor Sharing Flows in the Internet. Thirteenth International Workshop on Quality of Service (IWQoS), June 2005.

[5] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, 1(4):397–413, August 1993.

[6] S. Floyd, M. Handley, J. Padhye and J. Widmer. Equation-Based Congestion Control for Unicast Applications. Sigcomm Symposium on Communications Architectures and Protocols, 2000.

[7] S. Floyd. HighSpeed TCP for large congestion windows. Internet draft draft-floyd-tcp-highspeed-02.txt work in progress, `http://www.icir.org/floyd/hstcp.html`

[8] V. Jacobson. Congestion Avoidance and Control. *Proceedings of ACM Sigcomm*, 1988.

[9] R. Jain, W. Hawe, and D. Chiu. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report DEC TR-301, Digital Equipment Corporation, 1984.

[10] R. Jain. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks. *ACM Computer Communication Review*, 19(5):56–71, Oct. 1989.

[11] C. Jin, D. X. Wei, and S. H. Low. TCP FAST: motivation, architecture, algorithms, performance. In *Proceedings of IEEE Infocom*, March 2004.

[12] R. Johari and J. Tsitsiklis. Efficiency loss in a network resource allocation game. *Mathematics of Operations Research*,29(3): 407-435, 2004

[13] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high-bandwidth delay product networks. In *Proceedings of ACM Sigcomm*, August 2002.

[14] F. Kelly, A. Maoulloo, and D. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252,1998.

[15] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. Computer Communication Review 32(2), April 2003

[16] S. Kunniyur and R. Srikant. End-to-end congestion control: utility functions, random losses and ECN marks. *IEEE/ACM Transactions on Networking*, 11(5):689-702, Oct 2003.

[17] S. Low and D. Lapsley. Optimization flow control, I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6): 861-874, December 1999.

[18] S. Low. A duality model of TCP and queue management algorithms. *IEEE/ACM Trans. on Networking*, 11(4):525-536, August 2003.

[19] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556-567, Oct 2000.

[20] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, Internet Engineering Task Force, September 2001.

[21] T. Roughgarden and E. Tardos. How Bad Is Selfish Routing. *Journal of the ACM*, 49(2):236-259, 2002.

[22] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A compound tcp approach for high-speed and long distance networks. In *Proceedings of IEEE Infocom*, April 2006.

[23] A. Tang, J. Wang, S. Low and M. Chiang. Network Equilibrium of Heterogeneous Congestion Control Protocols. *Proceedings of IEEE Infocom*, 2005

[24] A. Tang, J. Wang, S. Hedge and S. Low. Equilibrium and fairness of networks shared by TCP Reno and Vegas/FAST. Telecommunication Systems, 30(4):417-439, December 2005.

[25] A. Tang, J. Wang, S. Low and M. Chiang Equilibrium of Heterogeneous Congestion Control: Existence and Uniqueness. To appear in *IEEE/ACM Transactions on Networking*, Oct 2007.

[26] WAN in Lab. `http://wil.cs.caltech.edu`.

[27] Z. Wang and J. Crowcroft. Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm. *ACM Computer Communications Review*, April 1992.

[28] D. Wei, C. Jin, S. Low and S. Hegde FAST TCP: motivation, architecture, algorithms, performance. *to appear in IEEE/ACM Transactions on Networking*, 2007.

[29] B. Wydrowski, L. H. Andrew, M. Zukerman. MaxNet: A Congestion Control Architecture for Scalable Networks. IEEE Communications Letters, 2003

[30] L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congestion Control for Fast Long-Distance Networks. Proceedings of IEEE Infocom, 2004.

[31] H. Yaiche, R. Mazumdar, and C. Rosenberg. A game theoretic framework for bandwidth allocation and pricing in broadband networks. *IEEE/ACM Transactions on Networking*, 8(5), October 2000.