TCP-Friendly CBR-Like Rate Control

Jie Feng and Lisong Xu
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115
Email: {jfeng, xu}@cse.unl.edu

***Abstract*—While the current definition of TCP friendliness has enabled a wide variety of traffic control protocols other than TCP, it still considerably restricts the design space of TCP-friendly traffic control protocols. For example, some multimedia streaming applications prefer a smooth sending rate on a time scale of minutes, however, a UDP flow maintaining a smooth sending rate on such a long time scale is naturally not TCP friendly by the current definition. In this paper, we first give a new class of TCP friendliness definitions, called Stochastic TCP Friendliness, which greatly expands the design space of TCP-friendly traffic control protocols, while still effectively maintaining the desired stability and fairness of the Internet. In particular, we propose stochastic TCP friendliness in usual stochastic order as a more appropriate design guideline for traffic control protocols, which intuitively ensures that a UDP flow is friendly to all competing TCP flows with any increasing utility function. Second, we develop a congestion control protocol, called TCP-Friendly CBR-Like Rate Control, which achieves a smooth sending rate on a time scale of minutes, and at the same time is stochastically TCP friendly in usual stochastic order in most network environments.**

I. Introduction

Multimedia streaming applications usually prefer a smooth sending rate in order to provide users with better user-perceived media quality. For example, a recent experimental study [6] shows that the relative importance of packet sending rate, the compound of sending rate variation and packet loss, and packet delay to the satisfaction of Skype users is approximately 46%, 53%, and 1%, respectively. That is, the user satisfaction strongly depends on the smoothness of the packet sending rate.

The existing TCP-friendly congestion control protocols [22] can achieve a smoother sending rate than TCP, while still being friendly to the competing TCP traffic. Specifically, a UDP flow[1] is TCP friendly [2], [10] (referred to as *deterministic TCP friendliness* or DTF), if it achieves the same or less average sending rate than a TCP flow in the same network environment on a *short* time scale of several round-trip times (RTTs) (i.e. milliseconds to seconds). Deterministic TCP friendliness allows a traffic control protocol to have a relatively smooth sending rate on a short time scale, while effectively maintaining the stability and fairness of the Internet.

While deterministic TCP friendliness has enabled a wide variety of traffic control protocols [22] other than TCP, it still considerably restricts the design space of traffic control protocols. As an example, some multimedia streaming applications prefer a smooth sending rate on a time scale of minutes, which is much longer than that provided by deterministic TCP friendliness. For instance, the same experimental study [6] shows that Skype users are more sensitive to the rate variation in a half-minute interval. However, a UDP flow maintaining a smooth sending rate on such a long time scale is *naturally* not deterministically TCP friendly.

For this reason, several protocols [23], [17], [21] consider a *relaxed* deterministic TCP friendliness which requires a UDP flow to maintain the same or less average sending rate than a TCP flow in the same network environment *only* on a long time scale but not on a short time scale. Thus, they can achieve a smooth sending rate on a long time scale. However, we find that this relaxed deterministic TCP friendliness may severely degrade the performance of competing TCP traffic, and in the extreme case, make the Internet *unstable*, as illustrated below.

A. Motivating Example

We use NS-2 to simulate a single bottleneck with 10 Mbps link capacity, 10 ms RTT, and 100-packet DropTail routers. TCP flows arrive as a Poisson process with an average rate of 30 flows per second, and the size of a TCP flow follows Pareto distribution with an average of 20 packets and a shape parameter of 1.5. There is also a single long-lived UDP flow. First, we run one simulation by adjusting the sending rate of the UDP flow according to TCP congestion control algorithm, and measuring its average sending rate, denoted by Y^{TCP}. Next, we run another group of simulations by setting the sending rate of the UDP flow to a constant-bit rate (CBR) Y^{CBR}, which varies from $0.50Y^{\mathrm{TCP}}$ to $0.95Y^{\mathrm{TCP}}$.

Fig. 1 shows the average response time of all TCP flows, where the response time of a TCP flow is defined as the time for the TCP flow to complete its transfer. We can see that, as Y^{CBR} approaches to Y^{TCP}, the average response time of TCP flows competing with the CBR-controlled UDP flow increases quickly, and becomes considerably longer than that of TCP flows competing with the TCP algorithm-controlled UDP flow. In the limiting case when $Y^{\mathrm{CBR}} = Y^{\mathrm{TCP}}$, it approaches to infinite, and the system becomes unstable.

B. Contributions of the Paper

In this paper, we consider the following two questions: First, *how to expand the design space of TCP-friendly traffic control*

The work presented in this paper is supported in part by NSF CAREER 0644080

[1]To simplify the discussion, we assume that all TCP-friendly congestion control protocols are implemented on top of UDP.

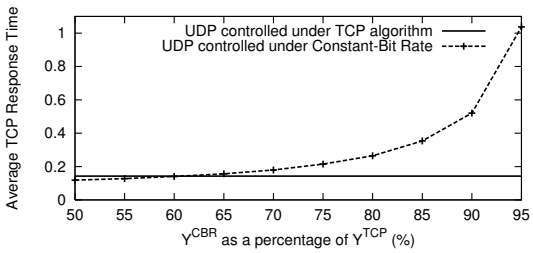

Fig. 1. As Y^{CBR} approaches to Y^{TCP}, the average TCP response time increases very quickly. In the limiting case when $Y^{\mathrm{CBR}} = Y^{\mathrm{TCP}}$, it approaches to infinite, and the system becomes unstable.

protocols, while still effectively maintaining the stability and fairness of the Internet? Second, *how to design a TCP-friendly congestion control protocol to achieve a smooth sending rate on a time scale much longer than that of the current TCP-friendly congestion control protocols?*

Correspondingly, the main contributions of our work are as follows. First, we give a new class of TCP friendliness definitions, called *Stochastic TCP Friendliness* (STF). Intuitively, stochastic TCP friendliness considers the statistical impact of UDP traffic on the rate distribution of all TCP flows, whereas deterministic TCP friendliness considers the specific impact of UDP traffic on the rate of each TCP flow. Several types of stochastic TCP friendliness are defined according to the size and type of the set of TCP utility functions. In particular, we propose stochastic TCP friendliness in usual stochastic order (STF-st) as a more appropriate design guideline for traffic control protocols, which intuitively ensures that a UDP flow is friendly to all competing TCP flows with any increasing utility function.

Second, we propose a congestion control protocol, called *TCP-Friendly CBR-Like Rate Control* (TFCBR), which is designed to achieve a CBR-like rate (in other words, a smooth sending rate) on a long time scale of minutes, and at the same time to be STF-st in most network environments. It is easy to achieve a long smoothness time scale, but it is very challenging to make TFCBR STF-st in most network environments. We use both flow-level network analysis and packet-level network simulation to determine the parameters of TFCBR, and finally the performance of TFCBR is evaluated with NS-2 simulation.

C. Organization of the Paper

The rest of the paper is organized as follows. Section II reviews the related work. Section III defines a new class of TCP friendliness definitions in the framework of stochastic TCP friendliness. Section IV describes the design goals and implementation details of the TFCBR protocol. Section V discusses how to set the TFCBR parameters to achieve its design goals. Section VI evaluates TFCBR performance with extensive NS-2 simulation. Finally, we conclude the paper by Section VII.

II. Related Work

Related Work on TCP Friendliness: Deterministic TCP-friendliness [2], [10] has been adopted in a large number of so called TCP-friendly congestion control protocols [22], such as TCP-Friendly Rate Control (TFRC) [11]. Several recent definitions of TCP friendliness consider the impact of UDP traffic on the overall performance of a stochastic workload of TCP flows, instead of the impact on the rate of each TCP flow. Key *et al.* [14] require that a TCP-friendly protocol should maintain the same average number of TCP flows in the network as the case when UDP traffic is controlled under TCP algorithm. Chiu and Tam [8], [7] study TCP friendliness with a network utility maximization model. Specifically, a protocol is considered to be better than another one, if it can achieve both higher overall TCP utility and higher overall UDP utility than the other one. Our proposed stochastic TCP friendliness is inspired by the work by Chiu and Tam. The main difference is that their work compares two protocols for a given TCP utility function, whereas our work compares two protocols for a given set of TCP utility functions in the framework of stochastic ordering [18], and then their statistical impacts on the rate distribution of all TCP flows can be compared according to the size and type of the set of TCP utility functions.

Differences between This Paper and Our Previous Work [9]: The notion of stochastic TCP friendliness was first described in [9], which defines a basic single-class stochastic TCP friendliness, whereas this paper defines a more general multiple-class stochastic TCP friendliness. More importantly, this paper proposes and studies STF-st, which is not considered in [9]. Finally, our previous work [9] proposes an admission control protocol, whereas this paper proposes a congestion control protocol.

Related Work on TCP-Friendly CBR-Like Rate Control: Since a UDP flow maintaining a smooth sending rate on a long time scale is naturally not deterministically TCP friendly, there is very little work related to TFCBR. Several congestion control protocols [23], [17], [21] achieve a smooth sending rate on a long time scale by considering the relaxed deterministic TCP friendliness. However, as demonstrated in Section I, the relaxed deterministic TCP friendliness is not friendly to TCP flows and may make the network unstable.

III. Framework of Stochastic TCP Friendliness

In this section, we give a new class of TCP friendliness definitions, called Stochastic TCP Friendliness (STF), which is designed to considerably expand the design space of TCP-friendly traffic control protocols than deterministic TCP friendliness, while still maintaining the desired stability and fairness in the Internet.

A. Definition

Stochastic TCP friendliness is defined in the framework of stochastic ordering [18], which is a well-established topic in probability theory to compare two distribution functions. We consider a network shared by a stochastic workload of TCP and UDP flows. We consider multiple classes of TCP flows generated by different types of TCP users. For example, short-lived TCP flows by web TCP users and long-lived TCP flows by peer-to-peer file-sharing TCP users. Let random variable

X_i^π denote the average sending rate of a TCP flow in class i when UDP flows are controlled under traffic control protocol π, and vector $\mathbf{X}^\pi$ denotes the TCP rate vector $(X_1^\pi, X_2^\pi, ...)$. As a reference protocol, we also consider the case when UDP flows are controlled under the TCP congestion control algorithm (referred to as TCP algorithm hereinafter), and use $\mathbf{X}^{\mathrm{TCP}}$ to denote the corresponding TCP rate vector. Stochastic TCP friendliness is defined as follows:

Definition 1: Protocol π is stochastically TCP-friendly (STF) with respect to TCP utility function set U, written

$$\mathbf{X}^{\mathrm{TCP}} \leq_U \mathbf{X}^\pi, \text{ if } \mathbb{E}[u(\mathbf{X}^{\mathrm{TCP}})] \leq \mathbb{E}[u(\mathbf{X}^\pi)] \quad \forall u \in U$$

where $\mathbb{E}[u(\mathbf{X}^{\mathrm{TCP}})] \leq \mathbb{E}[u(\mathbf{X}^\pi)]$ is component-wise comparison; that is, $\mathbb{E}[u(X_i^{\mathrm{TCP}})] \leq \mathbb{E}[u(X_i^\pi)]$ for $\forall i$. Intuitively, protocol π is stochastically TCP friendly, if the overall satisfaction of TCP users in each class measured with any utility function in U is higher when UDP flows are controlled by protocol π than that when UDP flows are controlled by TCP algorithm. Note that, this implies that the overall satisfaction of all TCP users in all classes is also higher with protocol π than that with TCP algorithm. For example, the overall satisfaction of web TCP users is higher, the overall satisfaction of peer-to-peer TCP users is higher, and the overall satisfaction of all TCP users is also higher. The relation $\leq_U$ gives a partial order that can be used to relatively compare the degrees of TCP friendliness of two protocols.

To make the definition flexible, we do not specify the type and number of TCP classes, and we leave it to the designer of a traffic control protocol to determine. This definition can be extended to the case where different TCP classes have different TCP utility function sets, and then stochastic TCP friendliness is defined with respect to a vector of TCP utility function sets, instead of a single TCP utility function set. This definition can also be extended to the case where the utility of a TCP flow is a function of other TCP performance metrics, such as a function of the TCP response time, or a function of both the average TCP rate and the TCP flow size [8], [7]. We leave these extensions for future research.

It is worth to point out that stochastic TCP friendliness considers a stochastic workload of TCP flows, whereas deterministic TCP friendliness usually considers a static workload of TCP flows (i.e. with only a fixed number of long-lived TCP flows). The reason is that stochastic workload is essential for us to study the TCP rate distribution and the network stability that refers to the fact that there exists a steady state for the network [16]. For example, the relaxed deterministic TCP friendliness is fair to the competing TCP flows and the network is stable in the case with a static workload of TCP flows. However, it is unfair to competing TCP flows and makes the network unstable in the case with a stochastic workload of TCP flows, as demonstrated in Section I.

B. TCP Utility Function Sets

A fundamental question is for what set of TCP utility functions stochastic TCP friendliness should be defined. Below, we consider three types of TCP utility function sets and correspondingly give three definitions of stochastic TCP friendliness. To simplify the description, we use increasing to refer to non-decreasing throughout the paper.

Definition 2: If U contains all increasing linear functions, written $\mathbf{X}^{\mathrm{TCP}} \leq_\mu \mathbf{X}^\pi$, then π is stochastically TCP-friendly in mean order (STF-μ).

If protocol π is STF-μ, then the overall average rate of all TCP flows in each class is higher when UDP traffic is controlled by protocol π than that when UDP traffic is controlled by TCP algorithm. Intuitively, *an STF-μ protocol is friendly to all TCP users interested in their average performance*.

Definition 3: If U contains all increasing concave functions, written $\mathbf{X}^{\mathrm{TCP}} \leq_{icv} \mathbf{X}^\pi$, then π is stochastically TCP-friendly in increasing concave order (STF-icv).

Since the user satisfaction of a file-transfer application is usually considered as an increasing concave function, the utility of a TCP flow is usually considered as an increasing concave function [20], [1]. Intuitively, *an STF-icv protocol is friendly to all TCP users with any increasing concave utility function, such as a usual file-transfer TCP user*.

Definition 4: If U contains all increasing functions, written $\mathbf{X}^{\mathrm{TCP}} \leq_{st} \mathbf{X}^\pi$, then π is stochastically TCP-friendly in usual stochastic order (STF-st).

TCP as the predominate transport protocol has been used in a wide variety of applications, and consequently there are a wide variety of TCP utility functions including both concave and non-concave functions. However, it is natural and reasonable to assume that the utility function of every TCP flow is an increasing function. Therefore, intuitively, *an STF-st protocol is friendly to all TCP users in the Internet*.

Fig. 2 shows the tradeoff between the impact of each STF type on the TCP rate distribution and the size of its TCP utility function set. We can see that *the larger the size of a TCP utility function set, the lower the impact of the corresponding stochastic TCP friendliness*. For example, two extreme cases are STF-st and NONE (i.e., no congestion control for UDP traffic). STF-st has the largest TCP utility function set (i.e., all increasing functions), and then has the lowest impact on the TCP rate distribution; whereas NONE has the smallest TCP utility function set (i.e., empty set), and then has the highest impact on the TCP rate distribution. Specifically, according to the theory of stochastic ordering [18], we have the following two theorems about the impact of an STF-μ protocol and an STF-st protocol on the TCP rate distribution.

Theorem 1: Protocol π is STF-μ (i.e. $\mathbf{X}^{\mathrm{TCP}} \leq_\mu \mathbf{X}^\pi$), if and only if $\mathbb{E}\left[X_i^{\mathrm{TCP}}\right] \leq \mathbb{E}\left[X_i^\pi\right]$ for $\forall i$.

Theorem 2: Protocol π is STF-st (i.e. $\mathbf{X}^{\mathrm{TCP}} \leq_{st} \mathbf{X}^\pi$), if and only if $F_i^{\mathrm{TCP}}(x) \geq F_i^\pi(x)$ for $\forall x$ and for $\forall i$, where cumulative distribution functions (CDFs) $F_i^{\mathrm{TCP}}(x) = \mathbb{P}\left[X_i^{\mathrm{TCP}} \leq x\right]$ and $F_i^\pi(x) = \mathbb{P}\left[X_i^\pi \leq x\right]$.

C. Relation between Different TCP Friendliness Definitions

If a TCP utility function set contains all increasing functions, then it also contains all increasing concave functions. Therefore, all STF-st protocols are STF-icv. Similarly, all

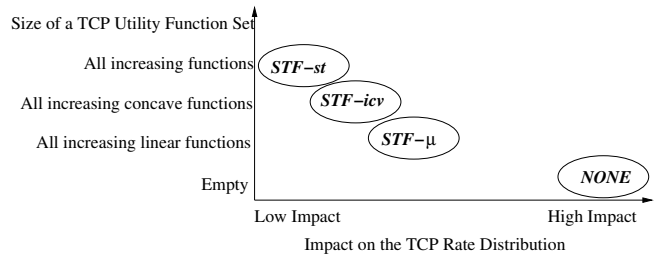

Fig. 2. The larger the size of a TCP utility function set, the lower the impact of the corresponding stochastic TCP friendliness on the TCP rate distribution.

STF-icv protocols are STF-μ. Deterministic TCP friendliness requires that a UDP flow obtains the same or less average rate than a TCP flow on a short time scale. In the limiting case when the time scale approaches to zero, a DTF protocol behaves exactly as TCP. Therefore, in the limiting case, all DTF protocols are STF-st. We have

$$\text{DTF (limiting case)} \subset \text{STF-st} \subset \text{STF-icv} \subset \text{STF-}\mu$$

where each of them denotes the set of corresponding protocols. The relaxed deterministic TCP friendliness requires that a UDP flow obtains the same or less average rate than a TCP flow only on a long time scale but not on a short time scale. In the limiting case when the time scale approaches to infinity, a relaxed DTF flow behaves exactly as a CBR flow. As analyzed in Section V and verified in Section VI, we have

$$\text{Relaxed DTF (limiting case)} \nsubseteq \text{STF-}\mu$$

IV. The TCP-Friendly CBR-Like Rate Control (TFCBR) Protocol

A. Design Goals

TCP-Friendly CBR-Like Rate Control (TFCBR) is designed for applications that last for a long time and prefer a long smoothness time scale, such as media streaming applications. TFCBR is designed as an end-to-end congestion control protocol that

- Goal 1: is stochastically TCP friendly in usual stochastic order in most network environments, and is stochastically TCP friendly in mean order in all network environments;
- Goal 2: achieves a smooth sending rate on a long time scale of minutes, whereas the current TCP-friendly congestion control protocols [22] maintain a smooth sending rate on a short time scale of milliseconds to seconds;
- Goal 3: maintains the bandwidth fairness among TFCBR flows; that is, every ongoing TFCBR flow has approximately the same instantaneous sending rate.

TFCBR is designed to be stochastically TCP friendly to the following two classes of TCP flows.

- TCP class 1 contains short-lived TCP flows, which stay in the network for a relatively short time period as compared to TFCBR flows. For example, web TCP connections, which usually last for seconds to minutes, whereas long-lived media streaming flows usually last for minutes to hours.
- TCP class 2 contains long-lived TCP flows, which stay in the network for a relatively long time period as compared to TFCBR flows. For example, long-lived FTP connections and long-lived TCP connections in peer-to-peer file-sharing applications, which may last for hours to days.

In most network environments, TFCBR ensures that the overall satisfaction of short-lived TCP users measured with any increasing utility function is higher when UDP flows are controlled under TFCBR than that when UDP flows are controlled under TCP algorithm, and the same holds for the overall satisfaction of long-lived TCP users and then holds for the overall satisfaction of both short-lived and long-lived TCP users together. In all network environments, TFCBR ensures that the overall average rate of short-lived TCP users is higher when UDP flows are controlled under TFCBR than that when UDP flows are controlled under TCP algorithm, and the same holds for the overall average rate of long-lived TCP users and then holds for the overall average rate of both short-lived and long-lived TCP users together.

B. TFCBR Parameters

To achieve the above design goals, TFCBR controls its sending rate with two parameters.

- Smoothness time scale β determines on what time scale TFCBR maintains a CBR-like sending rate, or in other words, a smooth sending rate.
- Relative sending rate α determines the ratio of the average rate of a TFCBR flow to the average rate of a TCP flow in the same network environment on the time scale of β.

The larger the value of parameter β, the longer the smoothness time scale of a TFCBR flow. However, there is a tradeoff between β and α (i.e. between the smoothness time scale and the average sending rate) in order for TFCBR to be STF-st, which will be discussed in the next section. In addition, parameter β determines the convergence speed of two TFCBR flows. The larger the value of parameter β, the longer the time for two TFCBR flows to converge to the bandwidth fair state.

C. TFCBR Implementation

TFCBR can be implemented by modifying only the sender side of any current TCP-friendly congestion control protocol [22], which can estimate the sending rate of a TCP flow in the same network environment. We choose to implement TFCBR by modifying TCP-Friendly Rate Control (TFRC) [11], since extensive network simulations and Internet experiments show that TFRC can estimate the TCP rate with a good accuracy in most network environments [11], [2].

TFRC has two states:

- Slow start. In this state, TFRC doubles its sending rate every RTT until the first packet loss, which is similar to the slow start state of TCP.
- Congestion avoidance. In this state, TFRC adjusts its sending rate every RTT based on the estimated TCP rate. TFRC calculates the weighted average loss event rate

based on the recent several loss intervals. It then estimates the TCP rate by using the TCP response function [19], which describes the TCP rate as a function of the packet size, round-trip time (RTT), loss event rate, timeout period, and the number of packets acknowledged by an ACK.

TFCBR also has two states:

- Slow start. TFCBR does not change the slow start part of TFRC.
- Congestion avoidance. TFCBR maintains all the TCP rates estimated by TFRC in the past β seconds. Every RTT, TFCBR sets its sending rate to the product of α and the average of all TCP rates estimated in the past β seconds. By doing so, the ratio of the average rate of a TFCBR flow to that of a TCP flow in the same network environment on the time scale of β is α, and TFCBR can maintain a CBR-like sending rate (or in other words, a smooth sending rate) on the time scale of β. During the time period after the slow start state and before TFCBR has collected a total of β seconds of estimated TCP rates, TFCBR sets its sending rate to the product of α and the average of all TCP rates estimated so far.

Another method that we have tried to implement TFCBR congestion avoidance state is to adjust TFCBR rate not every RTT but over a long time period, such as every β seconds. Even though this method can maintain a sending rate that is absolutely CBR within each time period, however, there might be a sharp rate increase or decrease between two consecutive time periods. Therefore, we do not chose this method.

There are some fundamental differences between TFRC and TFCBR. First, TFRC is designed according to the definition of deterministic TCP friendliness, which considers the specific impact of UDP traffic on the rate of each TCP flow. TFCBR is designed according to the definition of stochastic TCP friendliness, which considers the statistical impact of UDP traffic on the rate distribution of all TCP flows. Second, TFRC estimates the TCP rate in a short time period, which highly depends on the number of TCP flows in that short time period. TFCBR estimates the average TCP rate in the past β seconds. If β is sufficiently long, the estimated TCP rate mainly depends on the long-term TCP load instead of the instantaneous number of TCP flows. That is, TFCBR responds mainly to the long-term TCP load but not much on the instantaneous number of TCP flows. Therefore, TFCBR can maintain a smooth sending rate on a longer time scale than TFRC.

V. Setting TFCBR Parameters

This section discusses how to set TFCBR parameters α and β to achieve its design goals.

A. Guidelines for Setting TFCBR Parameters

The whole parameter space of TFCBR as illustrated in Fig. 3 is divided into three areas. The top area corresponds to points (α, β) so that TFCBR is not STF-μ, the middle area for STF-μ, and the bottom area for STF-st. We do not show the area of STF-icv in the figure, since TFCBR is designed to be STF-st in most cases, and to be STF-μ in the worst case.

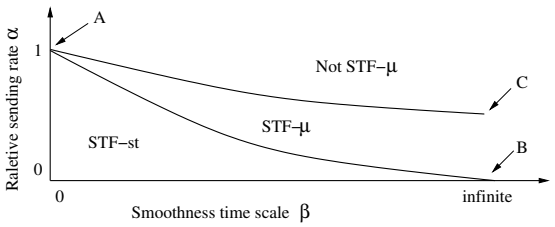

Fig. 3. The parameter space of TFCBR is divided into three areas according to the impact of α and β on TCP friendliness of TFCBR.

In order for TFCBR to be STF-st, its parameters should be selected so that the corresponding point (α, β) is in the STF-st area. The range of α for the STF-st area is [0,1]. At point A where $\alpha = 1$ and $\beta = 0$, TFCBR behaves exactly as TCP, and then it is both STF-st and STF-μ. Above point A, TFCBR is always more aggressive than TCP, and then it is neither STF-st nor STF-μ. The shape of the STF-st area shows the tradeoff between α and β. The larger the value of parameter α, the smaller the longest β for TFCBR to be STF-st.

If α is given, the longest β for TFCBR to be STF-st can be determined by finding the point with the given α and on the boundary line between the STF-st and STF-μ areas. However, this boundary line is variable and depends on the detailed statistic information of TCP traffic, but it is very hard for TFCBR as an end-to-end congestipn control protocol to accurately measure these information. Therefore, we have the following guideline.

- Guideline 1: To simplify the design of TFCBR, parameters α and β have fixed values, instead of variable values that require the detailed TCP statistic information.

However, a fixed point (α, β) and variable boundary lines imply that point (α, β) may sometimes in the STF-st area, but sometimes not in it. Consequently, we consider the following two design choices for α. 1) Parameter α should be no more than the α of point B, which is the boundary point between the STF-st and STF-μ areas when β approaches to infinity. 2) Parameter α should be no more than the α of point C, which is the boundary point between the STF-μ and non STF-μ areas when β approaches to infinity.

The first design choice ensures that for any β, TFCBR is always STF-st. Intuitively, this would be the best choice. However, as analyzed below in this section, the lower bound of the α of point B is zero. That is, only if the sending rate of TFCBR is zero, then for any β, TFCBR is always STF-st. Therefore, we do not choose the first design choice.

The second design choice ensures that for any β, TFCBR is either STF-st or STF-μ. That is, even with the variable boundary lines, we can still ensure that TFCBR is at least STF-μ. Therefore, we choose the second design choice for TFCBR, and the lower bound of the α of point C is analyzed below in this section. Following the second design choice, we have the following guidelines to achieve design goal 1.

- Guideline 2: Parameter α should be no more than the α of point C, so that TFCBR is STF-μ for any β.
- Guideline 3: Point (α, β) should be in the STF-st area in most cases, so that TFCBR is STF-st in most cases.

In addition, we have the following guidelines to achieve design goals 2 and 3.

- Guideline 4: Parameter β should be long enough for TFCBR to achieve a smooth sending rate on a time scale of minutes.
- Guideline 5: Parameter β should not be too large so that two TFCBR flows with different rates can converge to the fair share point at a reasonably fast speed.

Below, we first study the TFCBR parameter space using a flow-level network model in Section V-B, and then study the TFCBR parameter space using packet-level network simulations in Section V-C. Finally, we set the TFCBR parameters α and β in Section V-D.

B. Flow-Level Analysis of TFCBR Parameter Space

1) Flow-Level Network Model: To study the TFCBR parameter space, we use a flow-level network model where the bandwidth sharing among a dynamic number of competing TCP flows is analyzed by using a Processor-Sharing (PS) queueing model. The choice to use the flow-level network model is based on the following two reasons. 1) The flow-level network model has already been widely used to study the performance of TCP flows [13], [15]. It has been shown [13] that the flow-level analysis can accurately predict the essential properties observed in detailed packet-level TCP simulations, while not involving the complexity at the packet level. 2) Since we are interested in only the stochastic ordering of different traffic control protocols for UDP flows, we use the flow-level analysis only to *relatively* compare the TCP rate distributions under different protocols, *not* to calculate the *accurate* TCP rate distributions.

Our model considers a single bottleneck with a unit of link capacity shared by the following three types of flows.

- UDP: A fixed number of long-lived UDP flows, which stay in the network forever. Let n_u denote the number of long-lived UDP flows.
- TCP Class 1: A dynamic number of short-lived TCP flows, which stay in the network for a short time period. We assume that short-lived TCP flows arrive as a Poisson process with an average arrival rate of λ_1, and each has a random flow size with mean $1/\mu_1$. The load of short-lived TCP flows is then $\rho_1 = \lambda_1/\mu_1$.
- TCP Class 2: A fixed number of long-lived TCP flows, which stay in the network forever. Let n_2 denote the number of long-lived TCP flows.

We consider the following two protocols for UDP flows.

- TFCBR. In this case, all long-lived UDP flows are controlled under TFCBR. If the instantaneous sending rate of all UDP flows is r, and there are a total of n_1 short-lived TCP flows, then every TCP flow (short-lived or long-lived) has an instantaneous sending rate of $(1-r)/(n_1+n_2)$. Let random variable X_i^{TFCBR} denote the average sending rate of a TCP flow in class i when UDP flows are controlled under TFCBR, and $\mathbf{X}^{\mathrm{TFCBR}}$ denote TCP rate vector $(X_1^{\mathrm{TFCBR}}, X_2^{\mathrm{TFCBR}})$.
- TCP Algorithm. In this case, all long-lived UDP flows are controlled under the TCP congestion control algorithm. This case is used as a reference protocol to study the TCP friendliness of TFCBR. In this case, every flow (TCP or UDP) has an instantaneous sending rate of $1/(n_1+n_2+n_u)$. Let random variable X_i^{TCP} denote the average sending rate of a TCP flow in class i when UDP flows are controlled under TCP algorithm, and $\mathbf{X}^{\mathrm{TCP}}$ denote TCP rate vector $(X_1^{\mathrm{TCP}}, X_2^{\mathrm{TCP}})$.

2) Point B: In theory, point B can be determined by using Theorem 2. However, it is analytically intractable to obtain TCP rate distributions $\mathbb{P}\left[X_1^{\mathrm{TCP}} \le x\right]$ and $\mathbb{P}\left[X_1^{\mathrm{TFCBR}} \le x\right]$. Therefore, we estimate point B by calculating TCP number distributions $\mathbb{P}\left[N_i^{\mathrm{TCP}} \le n\right]$ and $\mathbb{P}\left[N_i^{\mathrm{TFCBR}} \le n\right]$, where random variables N_i^{TCP} and N_i^{TFCBR} denote the total number of TCP flows of class i in the network when UDP flows are controlled under TCP algorithm and TFCBR, respectively. We have the following theorem.

Theorem 3: For a bottleneck link with a unit of link capacity shared by n_u long-lived UDP flows, n_2 long-lived TCP flows, and a dynamic number of short-lived TCP flows arriving as a Poisson process with load $\rho_1 = \lambda_1/\mu_1$, TFCBR with $\beta = \infty$ achieves both $\mathbb{P}\left[N_1^{\mathrm{TCP}} \le n\right] \le \mathbb{P}\left[N_1^{\mathrm{TFCBR}} \le n\right]$ and $\mathbb{P}\left[N_2^{\mathrm{TCP}} \le n\right] \le \mathbb{P}\left[N_2^{\mathrm{TFCBR}} \le n\right]$ for any n, n_u, n_2 and ρ_1, if and only if $\alpha = 0$.

Proof: Since $N_2^{\mathrm{TFCBR}} = N_2^{\mathrm{TCP}} = n_2$, we consider only N_1^{TFCBR} and N_1^{TCP}.

First, we calculate $\mathbb{P}\left[N_1^{\mathrm{TFCBR}} \le n\right]$, when UDP flows are controlled under TFCBR. If $\beta = \infty$, TFCBR becomes a CBR flow, and let r denote the total sending rate of all UDP flows. Fig. 4 shows that the bottleneck link is modeled as an M/G/1-PS queueing node with service rate $1-r$ shared by a dynamic number of customers (i.e. short-lived TCP flows) with load ρ_1 and n_2 permanent customers (i.e. long-lived TCP flows), which has been analyzed in [3]. We have

$$\mathbb{P}\left[N_1^{\mathrm{TFCBR}} \le n\right] = \sum_{i=0}^{n} \left(1 - \frac{\rho_1}{1-r}\right)^{n_2+1} \binom{i+n_2}{n_2} \left(\frac{\rho_1}{1-r}\right)^{i} \quad (1)$$

Next, we calculate $\mathbb{P}\left[N_1^{\mathrm{TCP}} \le n\right]$, when UDP flows are controlled under TCP algorithm. Fig. 5 shows that the bottleneck link is modeled as an M/G/1-PS queueing node with a unit of service rate shared by a dynamic number of customers (i.e. short-lived TCP flows) with load ρ_1 and n_2+n_u permanent customers (i.e. long-lived TCP flows and long-lived UDP flows). We have

$$\mathbb{P}\left[N_1^{\mathrm{TCP}} \le n\right] = \sum_{i=0}^{n} (1-\rho_1)^{n_2+n_u+1} \binom{i+n_2+n_u}{n_2+n_u} \rho_1^i \quad (2)$$

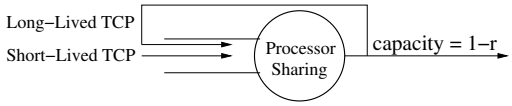

Fig. 4. When a UDP flow is controlled under TFCBR, the bottleneck link is modeled as an M/G/1-PS queueing node with link capacity $1-r$ shared by a dynamic number of customers (i.e. short-lived TCP flows) with load ρ_1 and n_2 permanent customers (i.e. long-lived TCP flows)

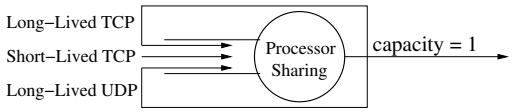

Fig. 5. When a UDP flow is controlled under TCP algorithm, the bottleneck link is modeled as an M/G/1-PS queueing node with a unit of link capacity shared by a dynamic number of customers (i.e. short-lived TCP flows) with load ρ_1 and n_2+n_u permanent customers (i.e. long-lived TCP flows and long-lived UDP flows).

After some cumbersome but straightforward manipulation, it can be shown that inequality $\mathbb{P}\left[N_1^{\mathrm{TCP}} \le n\right] \le \mathbb{P}\left[N_1^{\mathrm{TFCBR}} \le n\right]$ holds for any n, n_u, n_2 and ρ_1, if and only if $r=0$; that is $\alpha=0$. ∎

Therefore, we believe that TFCBR is STF-st for any β, only if the sending rate of TFCBR is zero. Consequently, we do not attempt to design TFCBR to be STF-st for any β.

3) Point C: In theory, point C can be determined by using Theorem 1. We are able to obtain average TCP rates $\mathbb{E}\left[X_2^{\mathrm{TCP}}\right]$ and $\mathbb{E}\left[X_2^{\mathrm{TFCBR}}\right]$. However, it is analytically intractable [15] to obtain average TCP rates $\mathbb{E}\left[X_1^{\mathrm{TCP}}\right]$ and $\mathbb{E}\left[X_1^{\mathrm{TFCBR}}\right]$. Therefore, we estimate point C by calculating a frequently used approximation [15] to the average rate. Let R_1^{TCP} and R_1^{TFCBR} denote the ratio of the average TCP flow size to the average TCP response time for class 1 when UDP flows are controlled under TCP algorithm and TFCBR, respectively. We have the following theorem.

Theorem 4: For a bottleneck link with a unit of link capacity shared by n_u long-lived UDP flows, n_2 long-lived TCP flows, and a dynamic number of short-lived TCP flows arriving as a Poisson process with load $\rho_1=\lambda_1/\mu_1$, TFCBR with $\beta=\infty$ achieves both $R_1^{\mathrm{TCP}} \le R_1^{\mathrm{TFCBR}}$ and $\mathbb{E}\left[X_2^{\mathrm{TCP}}\right] \le \mathbb{E}\left[X_2^{\mathrm{TFCBR}}\right]$ for any $n_u \ge 1$, $n_2 \ge 0$ and $\rho_1 \ge 0$, if and only if $\alpha \le 0.5$.

Proof: First, when UDP flows are controlled under TFCBR, Fig. 4 shows that the queueing model for the bottleneck link, which has been analyzed in [3]. We have $R_1^{\mathrm{TFCBR}} = (1-r-\rho_1)/(n_2+1)$, and $\mathbb{E}\left[X_2^{\mathrm{TFCBR}}\right] = X_2^{\mathrm{TFCBR}} = (1-r-\rho_1)/n_2$.

Next, when UDP flows are controlled under TCP algorithm, Fig. 5 shows the queueing model for the bottleneck. We have $R_1^{\mathrm{TCP}} = (1-\rho_1)/(n_2+n_u+1)$, and $\mathbb{E}\left[X_2^{\mathrm{TCP}}\right] = X_2^{\mathrm{TCP}} = (1-\rho_1)/(n_2+n_u)$.

After some cumbersome but straightforward manipulation, it can be shown that both $R_1^{\mathrm{TCP}} \le R_1^{\mathrm{TFCBR}}$ and $\mathbb{E}\left[X_2^{\mathrm{TCP}}\right] \le \mathbb{E}\left[X_2^{\mathrm{TFCBR}}\right]$ hold, if and only if

$$r \le \frac{n_u}{n_2+n_u+1}\left(1-\rho_1\right) \tag{3}$$

Let Y^{TFCBR} and Y^{TCP} denote the average sending rate of a UDP flow when it is controlled under TFCBR and TCP algorithm, respectively. We have $Y^{\mathrm{TFCBR}} = r/n_u$ and $Y^{\mathrm{TCP}} = (1-\rho_1)/(n_2+n_u)$. Therefore, we have

$$\alpha = \frac{Y^{\mathrm{TFCBR}}}{Y^{\mathrm{TCP}}} \le \frac{n_2+n_u}{n_2+n_u+1} \tag{4}$$

It is easy to see that Inequality 4 holds for any $n_u \ge 1$, $n_2 \ge 0$ and $\rho_1 \ge 0$ if and only if $\alpha \le 0.5$. ∎

Therefore, we believe the lower bound of the α of point C is 0.5. Consequently, according to guideline 2, TFCBR parameter α should be no more than 0.5 so that TFCBR is STF-μ for any β.

4) Stability Conditions: It is important to note the different stability conditions of the two queueing models. First, when UDP flows are controlled under TCP algorithm, the queueing model shown in Fig. 5 is stable if and only if $\rho_1 < 1$ [4], which is independent of the load of UDP traffic due to the adaptive nature of TCP algorithm. Second, when a UDP flow is controlled under TFCBR, the queueing model shown in Fig. 4 is stable if and only if $\rho_1 < 1-r$, which however depends on the total rate of UDP flows. Note that, $\rho_1 < 1-r$ is equivalent to $\alpha < 1+n_2/n_u$. When $n_2=0$, we have $\alpha<1$. Intuitively, this implies *in order to make the Internet stable, the average rate of a UDP flow with a long smoothness time scale must be less than that of a TCP flow in the same network environment, when there is no competing long-lived TCP flows.* This explains why and when the relaxed deterministic TCP friendliness discussed in Section I is unstable.

C. Packet-Level Simulation of TFCBR Parameter Space

To study the TFCBR parameter space, we also run packet-level NS-2 simulations. We simulate a dumbbell topology shared by short-lived Poisson TCP flows and a single long-lived UDP flow with parameters described in Section VI. TFCBR parameter α varies from 1/8 to 9/8, and parameter β varies from 1 seconds to 256 seconds.

We determine whether the corresponding TFCBR is STF-st, or STF-μ, or not STF-μ according to Theorem 2 and Theorem 1 using TFRC instead of TCP algorithm as a reference protocol. That is, we compare $\mathbf{X}^{\mathrm{TFCBR}}$ with $\mathbf{X}^{\mathrm{TFRC}}$ instead of $\mathbf{X}^{\mathrm{TCP}}$. The two main reasons that we choose TFRC instead of TCP algorithm as a reference protocol are as follows. 1) We are more interested in the impact of TFCBR parameters α and β, and less interested in the impact of the TFRC estimation error of the TCP rate. Using TFRC as a reference protocol can effectively eliminate the impact of TFRC estimation error, and help us focus on the impact of TFCBR parameters α and β. 2) TFRC is a reasonably good reference protocol, since extensive network simulations and Internet experiments show that TFRC is reasonably fair to competing TCP flows in most network environments [11], [2], and has been specified as a standard Internet standards track protocol [12].

Fig. 6, Fig. 7 and Fig. 8 show the simulation results with Droptail routers when the TCP load is 10%, 30% and 50% of the bottleneck capacity, respectively. Fig. 9, Fig. 10 and

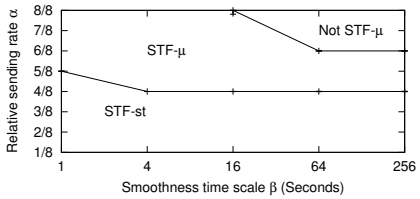

Fig. 6. TCP friendliness of TFCBR with TCP load=10% and DropTail routers.

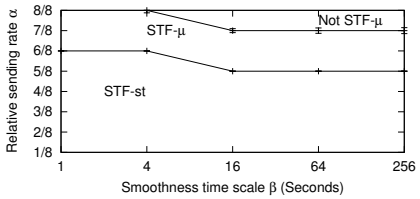

Fig. 7. TCP friendliness of TFCBR with TCP load=30% and DropTail routers.

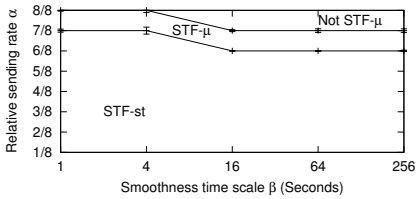

Fig. 8. TCP friendliness of TFCBR with TCP load=50% and DropTail routers.

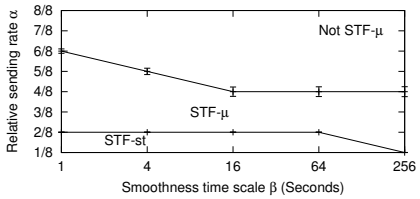

Fig. 9. TCP friendliness of TFCBR with TCP load=10% and RED routers.

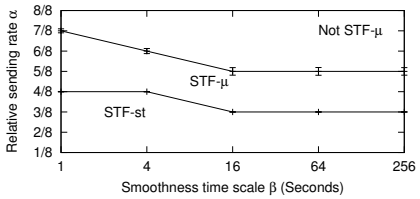

Fig. 10. TCP friendliness of TFCBR with TCP load=30% and RED routers.

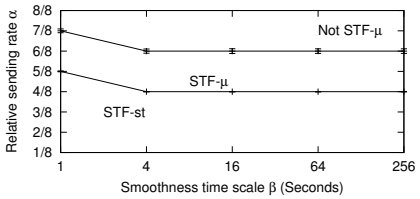

Fig. 11. TCP friendliness of TFCBR with TCP load=50% and RED routers.

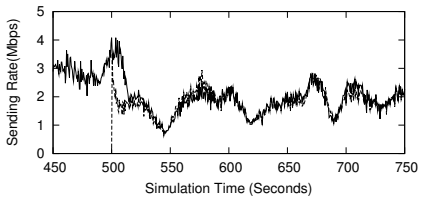

Fig. 12. Two TFCBR flows with β=16 seconds.

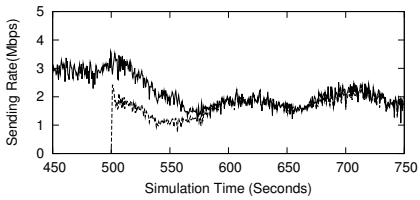

Fig. 13. Two TFCBR flows with β=64 seconds.

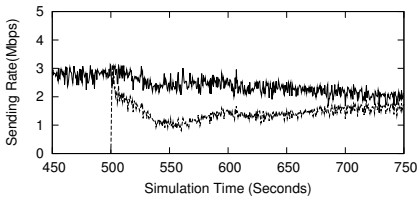

Fig. 14. Two TFCBR flows with β=256 seconds.

Fig. 11 show the simulation results with RED routers. We have the following observations. 1) The boundary lines are variable and depend on the network parameters, such as TCP load and queueing disciplines. 2) Among these figures, the largest α so that TFCBR is STF-μ for any β is 6/8 for Droptail routers and 4/8 for RED routers, which are very close to the asymptotical upper bound (i.e. 0.5) described in Theorem 4. 3) Among these figures, the largest α so that TFCBR is STF-st for any β is 4/8 for Droptail routers and 1/8 for RED routers, which are approaching to the asymptotical upper bound (i.e. 0) described in Theorem 3. 4) In some figures such as Fig. 6, TFCBR is still STF-μ even if $\alpha > 1$. Part of the reason is that the actual relative rate ratio (measured as the average TFCBR rate over the average TFRC rate in the same network environment) is not exactly equal to the specified value of α, especially with Droptail routers.

D. Setting TFCBR Parameters

According to guideline 1, we set α and β to fixed values. Since both flow-level analysis and packet-level simulation show that the lower bound of the α of point C is 0.5, we set α to be 0.5 according to guideline 2. Following guidelines 3, 4, and 5, we set β to 64 seconds.

Regarding guideline 3, we can see that in most cases point (0.5, 64) is in the STF-st area. Generally, when the TCP load is high, TFCBR with $\alpha = 0.5$ and $\beta = 64$ is STF-st and is then STF-μ. When the TCP load is low, TFCBR with $\alpha = 0.5$ and $\beta = 64$ is not STF-st but is still STF-μ. We believe this is reasonable, since intuitively TCP users are more interested in their performance when the traffic load is high, and less interested in their performance when the traffic load is low.

Regarding guidelines 4 and 5, TFCBR with $\beta = 64$ seconds can maintain a smooth sending rate on a time scale of minutes, and achieve a reasonably fast convergence speed. Below, we illustrate the tradeoff between smoothness time scale and convergence speed. We simulate a single bottleneck shared by short-lived Poisson TCP flows and two TFCBR flows. The load of short-lived Poisson TCP flows is set to 50% of the bottleneck capacity. The first TFCBR flow starts at second 10 and the second TFCBR flow starts at second 500. For both TFCBR flows, α is 0.5, and β varies from 16 seconds to 64 seconds, and 256 seconds. Fig. 12, Fig. 13, and Fig. 14 show their rates with $\beta = 16$, 64, and 256 seconds, respectively. We can clearly see that the larger the value of β, the smoother the TFCBR rate; however, the longer the convergence time.

VI. Simulation Result

A. Simulation Setup

Our NS-2 simulation uses a typical dumbbell topology. The bandwidth and one-way delay of the bottleneck link are set to 10Mbps and 10ms, respectively. Each source and sink are connected to the bottleneck link through different access links with random delays up to 0.5 ms to mitigate phase effect. The packet size is set to 1500 bytes for all connections. We test drop-tail routers with a buffer size of 40 packets. Every simulation runs for 75000 seconds.

We simulate three types of TCP flows. 1) Short-lived Poisson TCP flows that arrive as a Poisson process with an

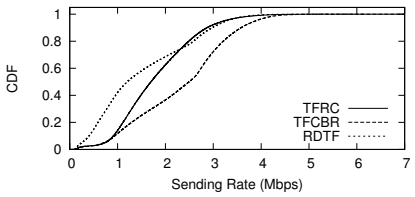

Fig. 15. The CDF of short-lived Poisson TCP rates when the TCP load is 30%.

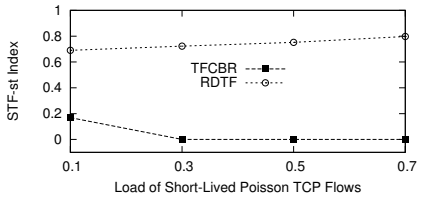

Fig. 16. STF-st index indicating the percentage of TCP flows whose rates should be improved in order to make the protocol STF-st.

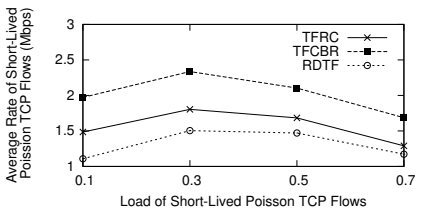

Fig. 17. Overall average rate of all short-lived Poisson TCP flows.

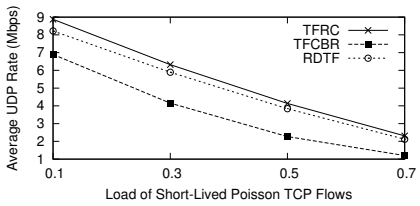

Fig. 18. Average rate of the UDP flow.

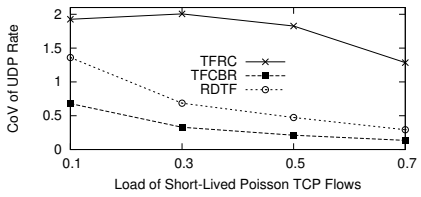

Fig. 19. CoV of the UDP flow measured on the time scale of 1 minute.

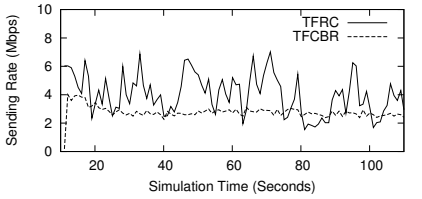

Fig. 20. The 1-second average UDP rate when the TCP load is 30%.

average arrival rate of λ_1 flows per second. The size of a TCP flow follows a Pareto distribution with average size $1/\mu_1$ =20 packets and shape parameter 1.6. In simulations, we vary the load of short-lived TCP flows that is defined as $\rho_1 = \lambda_1/\mu_1$. 2) Short-lived Web TCP flows that are generated by using PackMime [5], a source-level HTTP traffic model developed based on Internet traffic traces. Because of the limited space and because the results obtained with short-lived web TCP flows show the similar trends as the results obtained with short-lived Poisson TCP flows, we do not show the simulation results obtained with short-lived web TCP flows. 3) Long-lived TCP flows that stay in the network forever. In simulations, we vary the number of these long-lived TCP flows.

We simulate three traffic control protocols for UDP flows. 1) TFRC [11] as a representative protocol designed following the definition of Deterministic TCP Friendliness. For the same two reasons explained in Section V-C, we use TFRC as a reference protocol to study the degree of TCP friendliness of TFCBR. 2) TFCBR with $\alpha = 0.5$ and $\beta = 64$ seconds proposed in this paper following the definition of Stochastic TCP Friendliness. 3) TFCBR with $\alpha = 1$ and $\beta = 64$ seconds simulated as a representative protocol following the definition of Relaxed Deterministic TCP Friendliness. Below, they are referred to as TFRC, TFCBR, and RDTF, respectively. We determine whether TFCBR or RDTF is STF-st, or STF-μ, or not STF-μ according to Theorem 2 and Theorem 1 using TFRC as a reference protocol.

B. With Only Short-Lived Poisson TCP Flows

We simulate a single bottleneck shared by short-lived Poisson TCP flows and a single long-lived UDP flow. The load of short-lived TCP flows is changed from 10% to 70% of the bottleneck capacity. Fig. 15 shows the CDF of the TCP rates for each protocol when the TCP load is 30%. We can see that the CDF of TFCBR is always below that of TFRC, and then it is STF-st according to Theorem 2. However, a large fraction of the CDF of RDTF is above that of TFRC, and then it is not STF-st. Fig. 16 shows the STF-st index, which is defined as the fraction of the CDF of a protocol that is above the CDF of TFRC. The STF-st index of a protocol indicates the percentage of TCP flows whose rates should be improved to make the protocol STF-st. If the STF-st index of a protocol is zero, then it is STF-st; otherwise, the larger the STF-st index, the far away the protocol is to be STF-st. We can see that TFCBR is always STF-st, except when the TCP load is 10%. However, RDTF is not STF-st for any TCP load. Moreover, RDTF is far away to be STF-st in that about 80% of TCP flows should have an improved rate in order for RDTF to be STF-st. Fig. 17 shows the overall average rate of all TCP flows for each protocol. We can see that TFCBR is always STF-μ including the case when the TCP load is 10%, however, RDTF is not even STF-μ.

Fig. 18 shows the average sending rate of the UDP flow during the whole simulation, and Fig. 19 shows the coefficient of variation (CoV), which is calculated by first measuring the average UDP sending rate for each second, calculating the CoV of these rates within each 64-second interval, and finally taking the average of all obtained CoVs. These two figures show that TFCBR achieves a lower average sending rate than TFRC does since its α is 0.5, and maintains a much smoother sending rate on the time scale of 1 minute than TFRC does since its β is 64 seconds. In contrast, RDTF achieves a similar average sending rate as TFRC since its α is 1, and maintains a smoother sending rate on the time scale of 1 minute than TFRC does since its β is 64 seconds. Fig. 20 shows the 1-second average sending rates of TFRC and those of TFCBR when the TCP load is 30%. We do not show the rates of RDTF for the clarity of the figure. We can see that TFCBR first increases its rate very quickly in the slow start state, and then maintains a very smooth sending rate on a much longer time scale than TFRC in the congestion avoidance state.

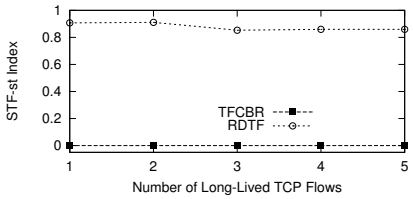

Fig. 21. STF-st Index based on the CDF of the short-lived TCP rates.

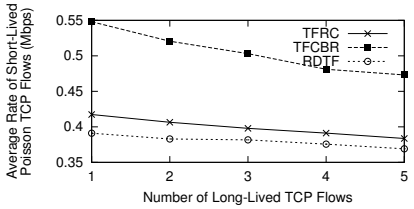

Fig. 22. Overall average rate of all short-lived Poisson TCP flows.

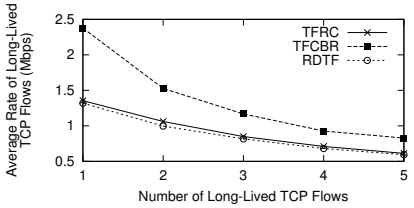

Fig. 23. Overall average rate of all long-lived TCP flows.

C. With both Short-Lived Poisson TCP flows and Long-Lived TCP Flows

We simulate a single bottleneck shared by short-lived Poisson TCP flows, long-lived TCP flows, and long-lived UDP flows. The load of short-lived Poisson TCP flows is set to 30% of the bottleneck capacity. The number of long-lived UDP flows is 5, and the number of long-lived TCP flows varies from 1 to 5. Fig. 21 shows the STF-st index based on the CDF of short-lived TCP rates, Fig. 22 shows the overall average rate of all short-lived TCP flows, and Fig. 23 shows the overall average rate of all long-lived TCP flows. We do not show the STF-st index for the CDF of long-lived TCP rates, since it can be derived from Fig. 23 due to the fact that all long-lived TCP flows have very similar average sending rates. For example, the overall average rate of all long-lived TCP flows with TFCBR is higher than that with TFRC for all cases, and then the CDF of long-lived TCP rates with TFCBR is always lower than that with TFRC for all cases. As a result, the STF-st index for the CDF of long-lived TCP rates with TFCBR is always zero. In addition, Fig. 21 shows that the STF-st index for the CDF of short-lived TCP rates with TFCBR is also always zero, and therefore TFCBR is always STF-st for any number of long-lived TCP flows. However, RDTF is not even STF-μ, since Fig. 22 shows that the overall average rate of all short-lived TCP flows with RDTF is lower than that with TFRC, and Fig. 23 shows that the overall average rate of all long-lived TCP flows with RDTF is also lower (even though slightly) than that with TFRC.

VII. Conclusions

This paper proposed a new class of TCP friendliness definitions called stochastic TCP friendliness, which greatly expands the design space of TCP-friendly traffic control protocols, while still effectively maintaining the desired stability and fairness of the Internet. This paper also developed TCP-Friendly CBR-Like Rate Control, which achieves a smooth sending rate on a time scale of minutes, and at the same time is stochastically TCP friendly in usual stochastic order in most network environments.

References

[1] M. Andrews, J. Cao, and J. McGowan. Measuring human satisfaction in data networks. In *Proceedings of IEEE INFOCOM*, Barcelona, Spain, April 2006.

[2] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker. Dynamic behavior of slowly-responsive congestion control algorithms. In *Proceedings of ACM SIGCOMM*, San Diego, CA, August 2001.

[3] J. Berg and O. Boxma. The M/G/1 queue with processor sharing and its relation to a feedback queue. *Queueing Systems: Theory and Applications*, 9(4):365–402, October 1991.

[4] T. Bonald and A. Proutière. On performance bounds for the integration of elastic and adaptive streaming traffic. In *Proceedings of ACM SIGMETRICS*, pages 235–245, New York, NY, June 2004.

[5] J. Cao, W. Cleveland, Y. Gao, K. Jeffay, F. Smith, and M. Weigle. Stochastic models for generating synthetic http source traffic. In *Proceedings of IEEE INFOCOM*, Hong Kong, March 2004.

[6] K. Chen, C. Huang, P. Huang, and C. Lei. Quantifying Skype user satisfaction. In *Proceedings of ACM SIGCOMM*, Pisa, Italy, September 2006.

[7] D. Chiu and A. Tam. Network fairness for heterogeneous applications. In *Proceedings of ACM SIGCOMM ASIA Workshop*, Beijing, China, April 2005.

[8] D. Chiu and A. Tam. Fairness of traffic controls for inelastic flows in the Internet. *Computer Networks Journal*, 51(11):2938–2957, August 2007.

[9] J. Feng and L. Xu. On the time scales of TCP-friendly admission control protocols. In *Proceedings of IEEE ICC*, Beijing, China, May 2008.

[10] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[11] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM SIGCOMM*, pages 43–56, Stockholm, Sweden, August 2000.

[12] S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP friendly rate control (TFRC): Protocol specification. *RFC 3448*, January 2003.

[13] S. Fredj, T. Bonald, A. Proutiere, G. Regnie, and J. Roberts. Statistical bandwidth sharing: A study of congestion at flow level. In *Proceedings of ACM SIGCOMM*, San Diego, CA, August 2001.

[14] P. Key, L. Massoulie, A. Bain, and F. Kelly. Fair internet traffic integration: network flow models and analysis. *Annals of Telecommunications*, 59, 2004.

[15] A. Kherani and A. Kumar. Stochastic models for throughput analysis of randomly arriving elastic flows in the Internet. In *Proceedings of IEEE INFOCOM*, New York City, NY, July 2002.

[16] L. Kleinrock. *Queueing Systems, Volume I: Theory*. John Wiley and Sons, New York, 1975.

[17] Y. Lai and S. Chien. A TCP-friendly congestion control to guarantee smoothness by slack term. In *Proceedings of IEEE ICCCN*, Chicago, IL, October 2004.

[18] A. Müller and D. Stoyan. *Comparison Methods for Stochastic Models and Risks*. John Wiley & Sons, Ltd, England, 2002.

[19] J. Padhye, V. Firoiu, D. Towsley, and J. Kursoe. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM*, pages 303–314, 1998.

[20] S. Shenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176–1188, September 1995.

[21] Z. Wang, S. Banerjee, and S. Jamin. Media-friendliness of a slowly-responsive congestion control protocol. In *Proceedings of ACM NOSSDAV*, Kinsale, Ireland, June 2004.

[22] J. Widmer, R. Denda, and M. Mauve. A survey on TCP-friendly congestion control. *IEEE Network Magazine*, 15(3):28–37, May 2001.

[23] J. Yan, K. Katrinis, M. May, and B. Plattner. Media and TCP friendly congestion control for scalable video streams. *IEEE Transactions on Multimedia*, 8(2), April 2006.