Understanding CHOKe: Throughput and Spatial Characteristics

Ao Tang Jiantao Wang Steven H. Low
California Institute of Technology, Pasadena, CA91125
{aotang@, jiantao@cds., slow@}caltech.edu

***Abstract*— A recently proposed active queue management, CHOKe, is stateless, simple to implement, yet surprisingly effective in protecting TCP from UDP flows. We present an equilibrium model of TCP/CHOKe. We prove that, provided the number of TCP flows is large, the UDP bandwidth share peaks at $(e+1)^{-1} = 0.269$ when UDP input rate is slightly larger than link capacity, and drops to zero as UDP input rate tends to infinity. We clarify the spatial characteristics of the leaky buffer under CHOKe that produce this throughput behavior. Specifically, we prove that, as UDP input rate increases, even though the total number of UDP packets in the queue increases, their spatial distribution becomes more and more concentrated near the tail of the queue, and drops rapidly to zero toward the head of the queue. In stark contrast to a nonleaky FIFO buffer where UDP bandwidth shares would approach 1 as its input rate increases without bound, under CHOKe, UDP simultaneously maintains a large number of packets in the queue and receives a vanishingly small bandwidth share, the mechanism through which CHOKe protects TCP flows.**

I. Introduction and Summary

TCP is believed to be largely responsible for preventing congestion collapse while the Internet has undergone dramatic growth in the last decade. Indeed, numerous measurements have consistently shown that more than 90% of traffic on the current Internet are still TCP packets, which, fortunately, are congestion controlled. Without a proper incentive structure, however, this state of affair is fragile and can be disrupted by the growing number of non-rate-adaptive (e.g., UDP-based) applications that can monopolize network bandwidth to the detriment of rate-adaptive applications. This has motivated several active queue management schemes, e.g., [7], [3], [4], [14], [9], [11], [2], that aim at penalizing aggressive flows and ensuring fairness. The scheme, CHOKe, of [11] is particularly interesting in that it does not require any state information and yet can provide a minimum throughput to TCP flows. In this paper, we provide an analytical model of CHOKe that explains both its throughput behavior and the spatial characteristics of its leaky buffer that underlies the throughput behavior.

The basic idea of CHOKe is explained in the following quote from [11]:

> When a packet arrives at a congested router, CHOKe draws a packet at random from the FIFO (first-in-first-out) buffer and compares it with the arriving packet. If they both belong to the same flow, then they are both dropped; else the randomly chosen packet is left intact and the arriving packet is admitted into the buffer with a probability that depends on the level of congestion (this probability is computed exactly as in RED).

The surprising feature of this extremely simple scheme is that it can bound the bandwidth share of UDP flows regardless of their arrival rate. Extensive simulation results in [11] show that as the arrival rate of UDP packets increases without bound, their bandwidth share peaks and then drops to zero! It seems intriguing that a flow that maintains a much larger number of packets in the queue does not receive a larger share of bandwidth, as in the case of a regular FIFO (first-in-first-out) buffer. A precise understanding of this phenomenon requires a detailed analysis of the queue dynamics.

We make two contributions in this paper. First, we present in Section II a deterministic fluid model that explicitly models both the feedback equilibrium of the TCP/CHOKe system and the spatial characteristics of the queue. By making three simplifying approximations to the model, we prove that, provided the number of TCP flows is large, the UDP bandwidth share peaks at $(e+1)^{-1} = 0.269$ when UDP input rate is slightly larger than link capacity, and drops to zero as UDP input rate tends to infinity (Theorems 1 and 2). This result, explained in Section III, has been independently obtained in [10] and [16] using different methods. It explains the simulation results of [11] and raises the question of what produces this throughput behavior.

Our second contribution answers this question by clarifying the spatial characteristics of the leaky buffer under CHOKe. In Section IV, we introduce the concepts of *spatial distribution* and *velocity* of packets at different positions in a queue. In a nonleaky FIFO buffer, both quantities are uniform across the queue. As a result, both the buffer occupancy of a flow and its bandwidth share are proportional to its input rate. CHOKe however produces a leaky buffer where packets may be dropped as they move toward the head of the queue, leading to *nonuniformity* in both quantities across the queue. We prove that, as UDP input rate increases, even though the total number of UDP packets in the queue increases, their spatial distribution becomes more and more concentrated near the tail of the queue, and drops rapidly to zero toward the head of the queue (Theorems 4 and 8). Hence, asymptotically, even though UDP packets occupy close to half of the queue (Theorem 5(a)), all of them are dropped before they advance to the head (Theorems 2 and 6). In stark contrast to a nonleaky FIFO buffer where UDP bandwidth shares would approach 1 as its input rate increases without bound, under CHOKe, UDP simultaneously maintains a large number of packets in the queue and receives a vanishingly small bandwidth share, the mechanism through which CHOKe protects TCP flows.

Our model can be solved numerically. The numerical solution, the throughput behavior and the spatial properties are accurately validated by the simulation results presented in Section V. We

To appear in IEEE/ACM Trans. on Networking, August 2004
Partial and preliminary results have appeared in [15], [16]. This paper is based on work supported by NSF through grants ANI-0113425 and ANI-0230967, AFOSR through grant F49620-03-1-0119, ARO through grant DAAD19-02-1-0283, and Cisco.

remark on some of our simulation experiences in Section VI and conclude in Section VII with limitations of this work.

II. Model

We focus on the single bottleneck FIFO buffer where packets are queued and drained at a rate of c packets per second. The buffer is shared by N identical TCP flows and a single UDP flow.[1] All TCP flows have a common round trip propagation delay of d seconds. We assume the system is stable and model its equilibrium behavior.

More generally, one can choose more than one packet from the queue, compare all of them with the incoming packet, and drop those from the same flow. This will improve CHOKe's performance, especially when there are multiple unresponsive sources [11], [10]. Here, we focus on the modeling of a single drop candidate packet. The analysis can be extended to the case of multiple drop candidates.

A. Notations

Quantities (rate, backlog, dropping probability, etc) associated with the UDP flow are indexed by 0. Those associated with TCP flows are indexed by $i = 1, \ldots, N$. Since the TCP sources are identical, these quantities all have the same value, and hence we will refer to flow 1 as the generic TCP flow. These are *equilibrium* quantities which we assume exist.

We collect here the definitions of all the variables and some of their obvious properties:

b_i: packet backlog from flow i, $i = 0, 1$.

b: total backlog; $b = b_0 + b_1 N$.

r: Congestion based dropping probability. The spatial properties of CHOKe are insensitive to the specific algorithm, such as RED, to compute this probability, as long as it is the same for all flows. In general, $r = g(b, \tau)$ for some function g as a function of aggregate backlog b and queueing delay τ.

h_i: The probability that an incoming packet of flow i, $i = 0, 1$, is dropped by CHOKe:

$$h_i = \frac{b_i}{b}$$

p_i: overall probability that a packet of flow i, $i = 0, 1$, is dropped before it gets through, either by CHOKe or RED:

$$p_i = 2h_i + r - rh_i \qquad (1)$$

The explanation of (1) is provided below.

x_i: source rate of flow i, $i = 0, 1$. The spatial properties of CHOKe are insensitive to the specific TCP algorithm, such as Reno or Vegas. In general, $x_1 = f(p_1, \tau)$ for some function f as a function of overall loss probability p_1 and queueing delay τ at equilibrium.

τ: common queueing delay. Round-trip time is $d + \tau$.

It is important to keep in mind that x_0 is the only independent variable; all other variables listed above are functions of x_0, though this is not made explicit in the notations. Later we will also use μ_i as a shorthand for flow i's normalized bandwidth share, $\mu_i := x_i(1 - p_i)/c$, $i = 0, 1$.

[1] In this paper, we use "UDP flow" to denote a flow with a constant rate.

B. TCP/CHOKe model

A packet may be dropped, either on arrival due to CHOKe or congestion (e.g., according to RED), or after it has been admitted into the queue when a future arrival from the same flow triggers a comparison. Let p_i be the probability that a packet from flow i is eventually dropped. To see why p_i is related to CHOKe and RED dropping probabilities according to (1), note that every arrival from flow i can trigger either 0 packet loss from the buffer, 1 packet loss due to RED, or 2 packet losses due to CHOKe. We assume that these events happen with respective probabilities of $(1 - h_i)(1 - r)$, $(1 - h_i)r$, and h_i. Hence, each arrival to the buffer is accompanied by an average packet loss of

$$2h_i + (1 - h_i)r + 0 \cdot (1 - h_i)(1 - r)$$

We take the overall loss probability p_i to be the packet loss rate $2h_i + (1 - h_i)r$. We now justify this probability from another perspective.

Consider a packet of flow i that eventually goes through the queue without being dropped. The probability that it is not dropped on arrival is $(1 - r)(1 - h_i)$. Once it enters the queue, it takes τ time to go through it. In this time period, there are on average τx_i packets from flow i that arrive at the queue. We assume that the probability that this packet is not chosen for comparison is

$$\left(1 - \frac{1}{b}\right)^{\tau x_i}$$

Hence, the overall probability that a packet of flow i survives the queue is

$$1 - p_i = (1 - r)(1 - h_i)\left(1 - \frac{1}{b}\right)^{\tau x_i} \qquad (2)$$

A simple interpretation of a leaky buffer is as follows: x_i is the source rate of flow i and $x_i(1-r)(1-h_i)$ is the rate at which flow i enters the queue after CHOKe and congestion-based dropping. This flow splits into two flows: one eventually exits the queue and the other is dropped inside the queue by CHOKe. The rate of the former flow is flow i's *throughput* $x_i(1 - p_i)$ and the rate of the latter flow is its *leak rate* $x_i h_i$, so that they sum to the input rate $x_i(1 - r)(1 - h_i)$. Since the link is fully utilized, the flow throughputs sum to link capacity:

$$x_0(1 - p_0) + N x_1(1 - p_1) = c$$

This completes the description of the model. In summary, the independent variable is UDP rate x_0. The ten dependent variables of the model are:

- backlogs b_i of flow i, $i = 0, 1$; total backlog $b = b_0 + N b_1$.
- congestion based dropping probability r, CHOKe dropping probabilities h_i, and overall dropping probabilities p_i, $i = 0, 1$.
- TCP rate x_1 and queueing delay τ.

The relations among these variables define our model. For ease of reference, we reproduce these ten equations here:

$$p_i = 2h_i + r - rh_i, \quad i = 0, 1 \tag{3}$$

$$p_i = 1 - (1-r)(1-h_i)\left(1 - \frac{1}{b}\right)^{\tau x_i}, \quad i = 0, 1 \tag{4}$$

$$h_i = \frac{b_i}{b}, \quad i = 0, 1 \tag{5}$$

$$b = b_0 + Nb_1 \tag{6}$$

$$c = x_0(1-p_0) + Nx_1(1-p_1) \quad \text{(full utilization)} \tag{7}$$

$$x_1 = f(p_1, \tau) \quad \text{(TCP)} \tag{8}$$

$$r = g(b, \tau) \quad \text{(e.g. RED)} \tag{9}$$

Let $z = (p_0, p_1, h_0, h_1, b_0, b_1, b, x_1, r, \tau)$ denote the ten dependent variables. Then the above equations (3)–(9) can be expressed as

$$F(z, x_0) = 0 \tag{10}$$

This can be regarded as implicitly defining z as a function of x_0.

We assume, in situations of interest:

A1: Given any $x_0 \geq 0$, there is a unique solution $z(x_0)$ that satisfies (10).

A2: Given any $x_0 \geq 0$, the solution $z(x_0)$ of (10) is continuous in x_0 and that $\lim_{x_0 \to \infty} z(x_0)$ exists. Denote $\lim_{x_0 \to \infty} z(x_0)$ by $z^\infty = (p_0^\infty, p_1^\infty, \ldots)$.

A3: The (equilibrium) TCP algorithm $f(p_1, \tau)$ is continuous in its arguments. Moreover, $x_1 = f(p_1, \tau) < \infty$ when $p_1 > 0$.

A4: The congestion based dropping $g(b, \tau)$ is continuous in its arguments. Moreover, $g(b, \tau) \to 1$ as $b \to \infty$.

Note that our model and analysis are insensitive to specifics of the algorithms (f, g) for TCP and congestion based dropping. They only need to satisfy conditions A3 and A4, which are nonrestrictive: A3 says that the TCP rate is finite if there is any loss, and A4 says that if backlog grows without bound then eventually all incoming packets will be dropped.

C. Numerical solution of TCP/CHOKe model

The set of nonlinear equations (3)–(9) that models the TCP/CHOKe system can be solved numerically by minimizing the quadratic cost (using (10)):

$$\min_z \quad J(z) \quad := \quad F(z, x_0)^T W F(z, x_0)$$

with an appropriate choice of positive diagonal weighting matrix W. A solution z^* of TCP/CHOKe satisfies $J(z^*) = \min_z J(z) = 0$. The solution can then be used in the differential equation model described below to solve for spatial properties of the leaky buffer under CHOKe; see Section IV.

Matlab is used to implement the above procedure. The weighting matrix is chosen such that each component in the vector $WF(z, x_0)$ is in the range [10 100] near the fixed point. A direct search method [8] for multidimensional unconstrained nonlinear minimization implemented in Matlab is used for this optimization problem. The search algorithm is stopped when $J(z)$ is smaller than 10^{-4}. The solution is accurately validated with ns-2 simulations; see Section V-A.

III. Throughput analysis

In this section, we make three approximations to our model. They allow us to readily derive the maximum achievable UDP throughput and a proof that UDP throughput approaches zero as $x_0 \to \infty$.

In the next section, we study the detailed dynamics of a leaky buffer that explains the mechanism underlying these macroscopic properties.

A. Three approximations

1) First approximation: Recall that an arrival packet is first subjected to CHOKe dropping, and if it survives CHOKe, then it is subjected to congestion based dropping (e.g., RED). First, we approximate the system by one in which the order of congestion based dropping and CHOKe is reversed: a packet is first admitted with probability $1 - r$, and if it is admitted, it is then compared with a packet randomly chosen from the queue and dropped with probability h_i.

With this approximation, the probability that a packet from flow i is eventually dropped is no longer given by (3), but instead,

$$p_i = 2h_i + r - 2rh_i \tag{11}$$

To see this, note that every arrival from flow i can trigger either 0 packet loss, 1 packet loss due to congestion, or 2 packet losses due to CHOKe. These events happen with respective probabilities of $(1-h_i)(1-r)$, r, and $(1-r)h_i$. Hence, each arrival to the buffer is accompanied by an average packet loss of

$$2(1-r)h_i + r + 0 \cdot (1-h_i)(1-r)$$

and hence the overall loss probability p_i in (11).[2] This implies that the probability that a packet of flow i goes through the queue without being dropped is:

$$1 - p_i = (1-r)(1-2h_i) \tag{12}$$

We now derive this probability from another perspective.

The same reasoning that leads to (2) applies here, except that τx_i is now replaced with $\tau x_i(1-r)$, so that the overall probability that a packet of flow i survives the queue is changed from (2) to:

$$1 - p_i = (1-r)(1-h_i)\left(1 - \frac{1}{b}\right)^{\tau x_i(1-r)} \tag{13}$$

Equating p_0 in (12) and (13), we have our first (of the three) key equation(s) to compute the maximum UDP throughput:

$$\frac{1-2h_0}{1-h_0} = \left(1 - \frac{1}{b}\right)^{\tau x_0(1-r)} \tag{14}$$

2) Second approximation: The second approximation is that N is so large that a comparison triggered by a *TCP* packet arrival never yields a match, i.e., we assume

$$h_1 = \frac{b_1}{b_0 + Nb_1} \leq \frac{1}{N} \simeq 0$$

This means that, once in the queue (after congestion based dropping and initial CHOKe), a TCP packet will never be dropped. The overall dropping probability p_1 then reduces to (substitute $h_1 = 0$ into (11)):

$$p_1 = r \tag{15}$$

[2] Note that the approximate probability (11) is smaller than the original loss probability given by (3) because a packet that is first dropped due to congestion saves a potential loss of two packets due to CHOKe. The difference however is small since both r and h_i are typically small.

More importantly, this provides a simple relation between queueing delay, throughput and backlog. From the condition (7) of full link utilization, the aggregate TCP throughput is $Nx_1(1-p_1) = c - x_0(1-p_0)$. The queueing delay is τ. The number of TCP packets in the buffer is $Nb_1 = b(1-h_0)$. Then Little's Theorem implies

$$\tau \quad = \quad \frac{Nb_1}{Nx_1(1-p_1)} \quad = \quad \frac{b(1-h_0)}{c-x_0(1-p_0)} \tag{16}$$

This is the second key equation for throughput analysis.

3) Third approximation: The third approximation is that the total backlog b is large so that

$$\left(1-\frac{1}{b}\right)^b \quad \simeq \quad e^{-1} \tag{17}$$

Combining the key equations (14), (16), and (17) to eliminate τ, we have

$$\frac{1-h_0}{1-2h_0} \quad = \quad \exp\left(\frac{x_0(1-r)(1-h_0)}{c-x_0(1-r)(1-2h_0)}\right) \tag{18}$$

where we have used (12) to eliminate p_0. This is the main equation in the proof of Theorem 1.

B. Maximum and asymptotic throughput

Let $\mu_0 = \mu_0(x_0)$ denote the UDP throughput share, $\mu_0 = x_0(1-p_0)/c$, and let $\mu_0^* = \max_{x_0 \geq 0} \mu_0(x_0)$ denote the maximum achievable UDP share. We now estimate μ_0^* and prove that μ_0 approaches 0 asymptotically as $x_0 \to \infty$. These results are also independently obtained in [10] (and [16]), using a different model.

Theorem 1. 1) *The maximum UDP bandwidth share is* $\mu_0^* = (e+1)^{-1} = 0.269$.
2) *It is attained when the UDP input rate after congestion based dropping is* $x_0^*(1-r^*) = c(2e-1)/(e+1) = 1.193c$.
3) *In this case, the CHOKe dropping rate for UDP is* $h_0^* = (e-1)/(2e-1) = 0.387$.

Proof. From (12), the UDP bandwidth share is

$$\mu_0 \quad = \quad (1-p_0)\frac{x_0}{c} \quad = \quad (1-r)(1-2h_0)\frac{x_0}{c}$$

Then rewrite (18) as

$$\frac{1-h_0}{1-2h_0} \quad = \quad \exp\left(\frac{1-h_0}{1-2h_0} \cdot \frac{\mu_0}{1-\mu_0}\right) \tag{19}$$

Let $\gamma(h_0)$ denote

$$\gamma(h_0) \quad := \quad \frac{1-h_0}{1-2h_0} \tag{20}$$

Then (19) becomes:

$$\gamma(h_0) \quad = \quad e^{\gamma(h_0)\frac{\mu_0}{1-\mu_0}}$$

or

$$\mu_0 \quad = \quad \frac{\ln\gamma(h_0)}{\gamma(h_0)+\ln\gamma(h_0)} \tag{21}$$

It is easy to check that the right-hand side has a unique maximum at $\gamma(h_0) = e$ with maximum bandwidth share μ_0^* given by

$$\mu_0 \quad \leq \quad \max_{\gamma} \frac{\ln\gamma}{\gamma+\ln\gamma} \quad = \quad \frac{1}{1+e} \quad =: \ \mu_0^* \tag{22}$$

Substituting $\gamma(h_0) = e$ into the definition (20) of $\gamma(h_0)$, the maximum UDP bandwidth share is attained when the CHOKe dropping probability h_0 for UDP is

$$h_0^* \quad = \quad \frac{e-1}{2e-1} \tag{23}$$

Since $h_0 = b_0/b$, this implies that 39% of the queue are UDP packets when UDP attains the highest throughput.

Since $\mu_0^* = x_0^*(1-r^*)(1-2h_0^*)/c$, the UDP rate after congestion based dropping, $x_0^*(1-r^*)$, that attains the maximum throughput is

$$x_0^*(1-r^*) \quad = \quad \frac{\mu_0^* c}{1-2h_0^*}$$

Substituting (22) and (23), we have

$$x_0^*(1-r^*) \quad = \quad \frac{2e-1}{e+1}c$$

□

The next result says that, as UDP rate x_0 grows without bound, even though UDP packets occupy up to half of the queue, its throughput $x_0(1-p_0)$ drops to zero. This result is also proved in Theorem 6 in Section IV-D using the original model (3)–(9) without the three approximations of this section.

Theorem 2. *As* $x_0 \to \infty$, $b_0 \to b/2$ *but* $\mu_0 \to 0$.

Proof. Since $p_0 \leq 1$, we have from (12)

$$h_0 \quad = \quad \frac{1}{2} \cdot \frac{p_0 - r}{1-r} \quad \leq \quad \frac{1}{2}$$

We argue that indeed $b_0/b = h_0 \to 1/2$ as $x_0 \to \infty$.

From (7), we have $x_0(1-p_0) \leq c$ for all $x_0 \geq 0$. Hence $p_0 \to 1$ as $x_0 \to \infty$. From (12), we have

$$(1-r^\infty)(1-2h_0^\infty) \quad = \quad 0$$

Hence either $r^\infty = 1$ or $h_0^\infty = 1/2$. We show $r^\infty < 1$ by contradiction. If $r^\infty = 1$, then $\lim_{x_0\to\infty} p_1 = r^\infty = 1$ by (15). Assumption A3 then implies $x_1^\infty = f(p^\infty, \tau^\infty) < \infty$. The condition of full link utilization (7) then implies

$$c \quad = \quad x_0^\infty(1-p_0^\infty) + Nx_1^\infty(1-p_1^\infty) \quad = \quad x_0^\infty(1-p_0^\infty)$$

This violates Theorem 1. Hence $r^\infty < 1$ and $h_0^\infty = 1/2$.

Then $\gamma \to \infty$ from (20), and hence, using (21), $\mu_0 \to 0$. □

We visualize equation (18) in Figure 1. It illustrates both theorems above.

C. Remarks: approximate model

With the first two approximations, the model (3)–(9) with ten dependent variables is simplified to eight dependent variables $(x_1, \tau, r, b, b_0, b_1, h_0, p_0)$, with $h_1 = 0$ and $p_1 = r$, and eight equations with the three equations (3)–(5) for $i = 1$ replaced by the single equation (16). The approximate model that consists of equations (3)–(5) for $i = 1$, (6)–(9) and (16) can also be solved numerically using the same method described in Section II-C.

We close this section by presenting another way to derive (14). The rate of flow i is $x_i(1-r)(1-h_i)$ when it first enters the tail of the queue after congestion-based dropping and CHOKe, and it takes τ seconds for packets to reach the head of the queue. After traveling down the queue for t seconds, $0 \leq t \leq \tau$, the packets

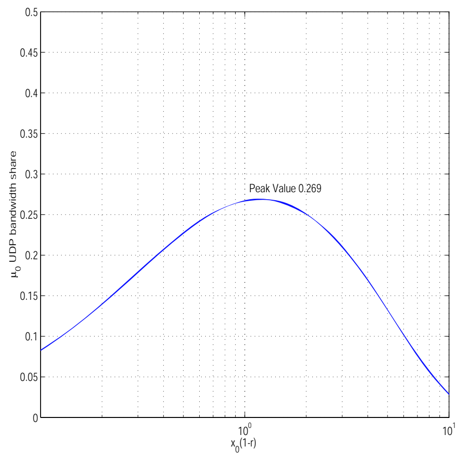


Fig. 1. μ_0 v.s. $x_0(1-r)$

arrive at a certain point $y(t) \in [0, b]$, where it has been thinned by a factor $(1-1/b)^{x_i(1-r)t}$ (following the same argument that leads to (2)) and the rate of the flow at $y(t)$ is

$$\tilde{x}_i(t) \quad := \quad x_i(1-r)(1-h_i)\left(1-\frac{1}{b}\right)^{x_i(1-r)t}$$

Hence $\tilde{x}_i(t)dt$ is the infinitesimal volume of the fluid at the point $y(t)$ in the queue, and the backlog from flow i is thus

$$b_i \quad = \quad \int_0^\tau x_i(1-r)(1-h_i)\left(1-\frac{1}{b}\right)^{x_i(1-r)t} dt$$

If we approximate $(1-1/b)^b$ by e^{-1}, when b is large, then the above integral reduces to

$$\frac{b_i}{b} \quad = \quad (1-h_i)\left(1-e^{-x_i(1-r)\tau/b}\right)$$

In particular, since $h_0 = b_0/b$, this implies

$$\frac{1-2h_0}{1-h_0} \quad = \quad e^{-x_0(1-r)\tau/b}$$

This is equivalent to (14) when we approximate $(1-1/b)^b$ by e^{-1}.

IV. Spatial characteristics

In this section, we derive the spatial characteristics of the leaky buffer under CHOKe that give rise to the macroscopic properties of maximum and asymptotic throughput proved in the last section.

A. Spatial distribution and packet velocity

If a packet cannot be dropped once it has been admitted into a FIFO queue, then, clearly, the queueing delay τ and bandwidth share μ_i are

$$\tau \; = \; \frac{b}{c} \qquad \text{and} \qquad \mu_i \; = \; \frac{b_i}{b} \tag{24}$$

For a leaky buffer where a packet can be dropped while it advances toward the head of the queue, (24) no longer holds, and the queueing delay and bandwidth share depend critically on the spatial characteristics of the queue. The key to their understanding is the spatial distribution of packets in the queue and the flow rate (velocity at which packets move through the queue) at different positions in the queue. We now define these two quantities and relate them to the variables previously defined.

Let $y \in [0, b]$ denote a position in the queue, with $y = 0$ being the tail and $y = b$ the head of the queue. In a leaky buffer, the queueing delay of a packet that *eventually* exits the queue is no longer the backlog it sees on arrival divided by the link capacity. This is because it advances toward the head both when packets in front of it exit the queue and when they are dropped by CHOKe. To model this dynamics, define $v(y)$ as the velocity at which the packet at position y moves toward the head of the queue:

$$v(y) \quad = \quad \frac{dy}{dt}$$

For instance, the velocity at the head of the queue equals the link capacity, $v(b) = c$. Then, the queueing delay τ is given in terms of $v(y)$ as

$$\tau \quad = \quad \int_0^\tau dt \quad = \quad \int_0^b \frac{1}{v(y)} dy \tag{25}$$

More generally, define, for $x_0 \geq 0$, $\tau(y)$ by

$$\tau(y) \quad = \quad \int_0^y \frac{1}{v(s)}\, ds \tag{26}$$

which can be interpreted as the time for a packet to reach position y from position 0. Clearly, $\tau(b) = \tau$.

Let $\rho_i(y)$ be the probability that the packet at position y belongs to flow i, $i = 0, 1$. As usual, we have

$$\rho_0(y) + N\rho_1(y) \quad = \quad 1, \qquad \text{for all } y \in [0, b] \tag{27}$$

The average number of flow i packets in the entire backlog satisfies

$$b_i \quad = \quad \int_0^b \rho_i(y) dy \tag{28}$$

More importantly, the bandwidth share μ_i is the probability that the head of the queue is occupied by a packet from flow i:

$$\mu_i \quad = \quad \rho_i(b) \tag{29}$$

Note that if the queue is not leaky, then the spatial distribution of packets will be uniform, $\rho_i(y)$ being independent of position y:

$$\rho_i(y) \quad = \quad \rho_i(b) \qquad \text{for all } y \in [0, b]$$

This, together with (28), implies the bandwidth share in (24), i.e., the bandwidth share depends only on the total number of flow i packets in the queue. When the queue is leaky, however, the spatial distribution can be highly non-uniform. The bandwidth share $\rho_i(b)$ of flow i depends on the spatial distribution of packets only at the *head* of the queue and does not depend directly on the distribution at other positions or the total number of flow i packets, in stark contrast to the case of nonleaky buffer. This is the underlying reason why UDP packets can occupy almost half of the queue, yet receiving very small bandwidth share: when UDP rate is high, $\rho_0(y)$ decreases rapidly from $y = 0$ to $y = b$ with $\rho_0(b) \simeq 0$; see Section IV-C.

We have completed the definition of spatial distribution $\rho_i(y)$ and velocity $v(y)$ of packets in the queue. We now derive an ordinary differential equation (ODE) model of these quantities.

B. ODE model of $\rho_i(y)$ and $v(y)$

We will derive an ODE model for $\rho_0(y)$ and $v(y)$; $\rho_1(y)$ can be obtained from (27).

Consider a small volume $v(y)dt$ of the (one-dimensional fluid) queue at position y. The amount of fluid (packets) in this volume that belongs to flow i is $\rho_i(y)v(y)dt$, $i = 0, 1$. For instance, $\rho_i(0)v(0)dt$, $i = 0, 1$, is the amount of fluid that arrives at the tail of the queue, packets that are not dropped by CHOKe or congestion based dropping on arrival and admitted into the buffer. Hence

$$\rho_i(0)v(0) \quad = \quad x_i(1-r)(1-h_i), \qquad i = 0, 1 \tag{30}$$

Another boundary condition is the packet velocity at the head of the queue mentioned above:

$$v(b) \quad = \quad c \tag{31}$$

Suppose the small volume $\rho_i(0)v(0)dt$ of fluid (our "tagged packet") arrives at the buffer at time 0, and reaches position y at time $\tau(y)$. During this period $[0, \tau(y)]$, there are $x_i\tau(y)$ packet arrivals from flow i, and each of these arrivals triggers a comparison. The tagged packet is selected for comparison with probability $1/b$ each time. We model this by saying that the fluid is thinned by a factor $(1-1/b)^{x_i\tau(y)}$ when it reaches position y at time $\tau(y)$. Thus

$$\rho_i(0)v(0)\left(1-\frac{1}{b}\right)^{x_i\tau(y)} \quad = \quad \rho_i(y)v(y) \tag{32}$$

Note that this is the same argument that leads to (2).

Taking logarithm on both sides and using (26) to eliminate $\tau(y)$, we have

$$\ln(\rho_i(y)v(y)) \quad = \quad \ln(\rho_i(0)v(0)) + \beta x_i \int_0^y \frac{1}{v(s)}\,ds$$

where

$$\beta \quad := \quad \ln\left(1-\frac{1}{b}\right)$$

Differentiating both sides with respect to y, we get

$$\beta\frac{x_0}{v(y)} \quad = \quad \frac{\rho_0'(y)}{\rho_0(y)} + \frac{v'(y)}{v(y)} \tag{33}$$

$$\beta\frac{x_1}{v(y)} \quad = \quad \frac{\rho_1'(y)}{\rho_1(y)} + \frac{v'(y)}{v(y)} \tag{34}$$

Now $(33) \times \rho_0(y) + (34) \times N \times \rho_1(y)$ yields

$$v'(y) \quad = \quad \beta\left(\rho_0(y)x_0 + (1-\rho_0(y))x_1\right) \tag{35}$$

where we have used (27). Substituting (35) into (33), we get

$$\rho_0'(y) = \beta(x_0 - x_1)\,\rho_0(y)(1-\rho_0(y))\,\frac{1}{v(y)} \tag{36}$$

Hence the spatial distribution $\rho_i(y)$ and packet velocity $v(y)$ is given by the two-dimensional system of nonlinear differential equations (35)–(36) with boundary conditions (30)–(31). Since the right-hand sides of (35)–(36) are continuously differentiable in (v, ρ_0), there exists a unique solution in its interval of existence [12].

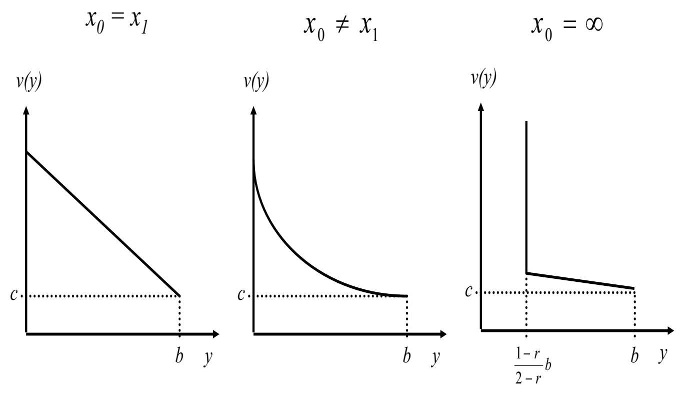
We make an important remark. Given x_0, quantities such as τ, b, b_i are uniquely determined by (10) by assumption A1. At the same time $v(y)$ and $\rho_0(y)$ are uniquely determined by the differential equations (35)–(36) with boundary conditions (30)–(31). The relations (25) and (28) between these two sets of quantities are not necessarily true a priori. Even though they seem reasonable based on their physical interpretation, they nonetheless remain a postulation:

A5: Relations (25) and (28) hold.

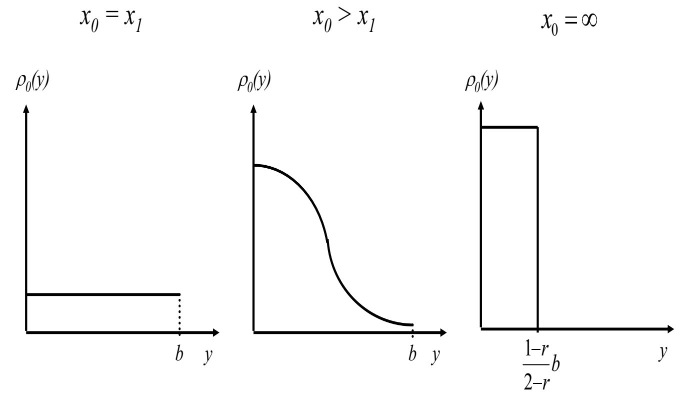
Note that (27) holds without assumption and defines $\rho_1(y)$.

C. Structural properties

In this subsection, we prove some structural properties of the velocity $v(y)$ and spatial distribution $\rho_0(y)$. They are illustrated in Figure 2.



(a) Velocity $v(y)$



(b) Spatial distribution $\rho_0(y)$

Fig. 2. Illustration of Theorems 3, 4, 6, 8 and 9

Theorem 3. *For all $x_0 \geq 0$, packet velocity $v(y)$ is a convex and strictly decreasing function with $v(0) = (x_0(1-h_0) + Nx_1(1-h_1))(1-r)$ and $v(b) = c$. It is linear if and only if $x_0 = x_1$.*

Proof. See Appendix VIII-A. □

Given x_0, define

$$\rho^* \quad := \quad \frac{1}{3}\left(1 - \frac{x_1}{x_0 - x_1}\right)$$

and the inflexion point y^* by

$$\rho_0(y^*) \quad = \quad \rho^*$$

Theorem 4. *Suppose $x_0 > x_1$. Then $\rho_0(y)$ is a strictly decreasing function, with*

$$\rho_0(0) = \frac{x_0(1-h_0)}{x_0(1-h_0) + Nx_1(1-h_1)}$$

Moreover,

- *if $\rho_0(0) \le \rho^*$, then $\rho_0(y)$ is convex.*
- *if $\rho_0(b) \ge \rho^*$, then $\rho_0(y)$ is concave.*
- *if $\rho_0(b) < \rho^* < \rho_0(0)$, then $\rho_0(y)$ is concave for $y \le y^*$ and convex for $y \ge y^*$.*

Proof. See Appendix VIII-B. □

Theorems 3 and 4 are illustrated in Figure 2. The figure also shows the asymptotic properties of $v(y)$ and $\rho_0(y)$, to which we now turn.

D. Asymptotic properties

We will prove that $v(y)$ and $\rho_0(y)$ take the form shown in the right-hand column of Figure 2 asymptotically as $x_0 \to \infty$. We assume:

A6: The pointwise limits of $v(y)$ and $\rho_0(y)$ as $x_0 \to \infty$, denoted by $v^\infty(y)$ and $\rho_0^\infty(y)$, exist. Moreover, relations (25), (26) and (28) are satisfied in the limit with $z(x_0)$, $v(y)$ and $\rho_0(y)$ replaced by $z^\infty(x_0)$, $v^\infty(y)$ and $\rho_0^\infty(y)$ respectively.

Note that we do not assume that the limit functions $v^\infty(y)$ and $\rho_0^\infty(y)$ satisfy the ODEs of Section IV-B.

We start with a result that says that regardless of the UDP rate x_0, every flow, including UDP flow, occupies less than half of the queue. This implies that asymptotically as $x_0 \to \infty$, congestion based dropping probability $r < 1$ and queueing delay $\tau > 0$. These properties are used later to prove the asymptotic UDP throughput and the asymptotic spatial properties of the leaky buffer of CHOKe.

Theorem 5. 1) *For all $x_0 \ge 0$, $b_i \le b/2$, $i = 0, 1$.*
2) *As $x_0 \to \infty$*
   a) *$h_0^\infty = (1-r)/(2-r)$ and $h_1^\infty = 1/N(2-r)$.*
   b) *$r^\infty < 1$.*
   c) *$x_1^\infty < \infty$ and $\tau^\infty > 0$.*
   d) *$b_i^\infty < \infty, i = 0, 1$*

Proof.

1) From (3), $2h_i + (1-h_i)r = p_i \le 1$ and hence using (5), we have
$$\frac{b_i}{b} = h_i \le \frac{1-r}{2-r}$$
The right-hand side is a decreasing function for $0 \le r \le 1$ with a maximum value of $1/2$ at $r = 0$.
2) Since $x_0(1-p_0) \le c$ for all x_0, p_0 must tend to 1 as $x_0 \to \infty$. Hence $2h_0 + (1-h_0)r = p_0 \to 1$. Then $h_0 \to (1-r)/(2-r)$, or $b_0 \to b(1-r)/(2-r)$. Since $b = b_0 + Nb_1$, we have $h_1 \to 1/N(2-r)$.
Suppose $r^\infty = 1$. From (3), $2h_i + (1-h_i)r = p_i \le 1$ and hence, for $i = 0, 1$,
$$r \le \frac{1-2h_i}{1-h_i} \le 1$$
with equalities if and only if $h_i = 0$ for $i = 0, 1$. But this contradicts that $h_0 + Nh_1 = 1$. Hence $r^\infty < 1$.
Consider TCP flow $i = 1$. Since $h_1^\infty = 1/(N(2-r^\infty)) > 0$ from part 2(a), (4) implies that $p_1^\infty > 0$. By assumption A3, $x_1^\infty = f(p_1^\infty, \tau^\infty) < \infty$. Now suppose for the sake of contradiction that $\tau^\infty = 0$. Then (4) implies
$$1 - p_1^\infty = (1-r^\infty)(1-h_1^\infty)$$
but (3) implies
$$1 - p_1^\infty = (1-r^\infty)(1-h_1^\infty) - h_1^\infty$$
yielding $h_1^\infty = 0$, contradicting that $h_1^\infty = 1/(N(2-r^\infty)) > 0$. Hence $\tau^\infty > 0$.
Finally, if $b_i^\infty = \infty$, then by assumption A4, we have $r^\infty = \lim_{x_0\to\infty} g(b_i, \tau) = 1$, contradicting (b).

□

We next show that the UDP throughput vanishes as x_0 grows without bound. This confirms the approximate throughput analysis of Theorem 2. Recall (26) that for $x_0 \ge 0$

$$\tau(y) = \int_0^y \frac{ds}{v(s)}.$$

Theorem 6. *As $x_0 \to \infty$, $x_0(1-p_0) = \rho_0(b)c \to 0$.*

Proof. From (36), we have

$$\frac{\rho_0'(y)}{\rho_0(y)(1-\rho_0(y))} = \beta(x_0 - x_1)\frac{1}{v(y)}$$

where $\beta = \ln(1 - 1/b)$. Integrating both sides from 0 to y, we get

$$\ln\frac{\rho_0(y)}{1-\rho_0(y)} - \ln\frac{\rho_0(0)}{1-\rho_0(0)} = \beta(x_0 - x_1)\tau(y)$$

Hence

$$\rho_0(y) = \frac{ae^{\beta(x_0-x_1)\tau(y)}}{1 + ae^{\beta(x_0-x_1)\tau(y)}} \quad (37)$$

where

$$a = \frac{\rho_0(0)}{1-\rho_0(0)} \quad (38)$$

UDP throughput share $x_0(1-p_0)/c$ is $\rho_0(y)$ evaluated at $y = b$ where $\tau(y) = \tau$. From Theorem 5(c), $\lim_{x_0\to\infty}\tau(b) = \tau^\infty > 0$, and hence by Lemma 7 below, $\lim_{x_0\to\infty}\rho_0(b) = 0$. □

The following lemma implies that, asymptotically as $x_0 \to \infty$, not only does the throughput of UDP $x_0(1-p_0) \to 0$, moreover, all UDP packets are dropped before the first point where $\tau(y)$ is nonzero. Note that y in the lemma below is generally a function of x_0.

Lemma 7. *If $\lim_{x_0\to\infty}\tau(y) > 0$, then*

$$\lim_{x_0\to\infty}\rho_0(y) = \lim_{x_0\to\infty}\frac{ae^{\beta(x_0-x_1)\tau(y)}}{1 + ae^{\beta(x_0-x_1)\tau(y)}} = 0$$

Proof. See Appendix VIII-C. □

The next result says that the asymptotic spatial distribution $\rho_0^\infty(y)$ of UDP takes the form shown in Figure 2.

Theorem 8. *Let $y^* := b^\infty(1-r^\infty)/(2-r^\infty)$. Then*

$$\rho_0^\infty(y) = \begin{cases} 1, & 0 \le y < y^* \\ 0, & y^* < y \le b^\infty \end{cases}$$

Proof. See Appendix VIII-D. □

The next result proves the shape of $v^\infty(y)$.

Theorem 9. *Let $y^* = b^\infty(1-r^\infty)/(2-r^\infty)$. Then*

$$v^\infty(y) = \begin{cases} \infty, & 0 \le y < y^* \\ c - \beta^\infty x_1^\infty\ (b^\infty - y), & y^* < y \le b^\infty \end{cases}$$

Proof. See Appendix VIII-E. □

We summarize these structural properties. First, when x_0 is large, the spatial distribution $\rho_0(y)$ decreases rapidly toward the head of the queue. This means that most of the UDP packets are dropped before they reach the head. It is therefore possible to simultaneously maintain a large number of packets (concentrating near the tail) and receive a small bandwidth share, in stark contrast to the behavior of a nonleaky FIFO buffer. Indeed, as x_0 grows without bound, UDP share drops to 0. Second, the packet velocity is infinite before the position y^* because UDP packets are being dropped at an infinite rate until y^*.

E. Asymptotic regime

The TCP/CHOKe system is much simpler in asymptotic regime when $x_0 = \infty$. To simplify notation, we drop the superscript on z^∞ in this subsection, though all quantities are limits.

By Theorem 6, UDP share $x_0(1-p_0)$ is zero, and $p_0 = 1$. Hence because of full utilization (7), TCP shares are equal:

$$x_1(1-p_1) = \frac{c}{N} \tag{39}$$

Using Theorem 5(2a) to eliminate h_1 in (3), we have

$$p_1 = \frac{2 + r(N(2-r)-1)}{N(2-r)} \tag{40}$$

Applying Theorem 9 to (25), we have

$$\tau = \frac{1}{-\beta x_1} \ln\left(1 - \frac{\beta x_1 b}{c(2-r)}\right) \tag{41}$$

where $\beta = \ln(1-1/b) < 0$. In summary, in asymptotic regime, the TCP/CHOKe model is reduced from ten dependent variables to five variables $z = (b, r, \tau, p_1, x_1)$, determined by the 5 equations (39)–(41), and (8), (9). Given the TCP function $f(p_1, \tau)$ in (8) and congestion based dropping $g(b, \tau)$ in (9), the system is completely specified and can be solved numerically.

The system is further simplified if we approximate $v^\infty(y) = c$ for $y > y^*$. Then (41) reduces to

$$\tau = \frac{b}{c(2-r)} \tag{42}$$

For TCP Reno TCP and RED, we use the following model for f and g (e.g., [5]):

$$p_1 = \frac{2}{2 + x_1^2(d+\tau)^2} \tag{43}$$

$$r = k(b - \underline{b}) \tag{44}$$

for some k and $\underline{b}$. With $c = 125$ pkts/s, $d = 0.1$s, $N = 32$, $\underline{b} = 20$ pkts, $k = 10^{-3}$, the asymptotic values are numerically solved to be (values in parentheses are calculated using (42) instead of (41)):

$$\begin{aligned} b &= 144.66 \ \ (143.92) \ \text{pkts} \\ r &= 0.125 \ \ (0.124) \\ \tau &= 0.6111 \ \ (0.6137) \ \text{s} \\ p_1 &= 0.1559 \ \ (0.1552) \\ x_1 &= 4.628 \ \ (4.624) \ \text{pkts/s} \end{aligned}$$

From ns-2 simulation with the same parameters at $x_0 = 100c$, we obtained $b = 143.15$ pkts, close to the numerical value.

V. Simulation results

We present three sets of simulation results. The first set illustrates the accuracy of our TCP/CHOKe model (3)–(9) and its macroscopic properties. The second set illustrates the spatial properties proved in Theorems 4 and 8. Both sets use only TCP NewReno. The third set uses TCP Vegas and illustrates that these properties are insensitive to the specific TCP algorithms.

We implemented a CHOKe module in ns-2 version 2.1b9 and have conducted extensive simulations using the network shown in Figure 3 to study the equilibrium behavior of CHOKe. There

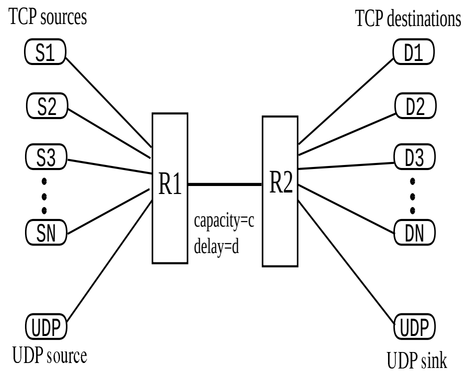


Fig. 3. Network topology

is a single bottleneck link from router R1 to router R2 shared by N TCP sources and one UDP source. The UDP source sends data at constant rate (CBR).

For all NewReno simulations, the link capacity is fixed at $c = 1$Mbps and the round trip propagation delay is $d = 100$ms. Packet size is 1KB. The simulation duration is 200–300 seconds. Parameters for Vegas simulations are given in Section V-C.

We use RED+CHOKe as the queue management with RED parameters: min_th $\underline{b} = 20$ packets, max_th $\overline{b} = 520$ packets, $p_{max} = 0.5$. The corresponding analytical model uses (43) for f and (44) for g, with $k = p_{max}/(\overline{b} - \underline{b})$.

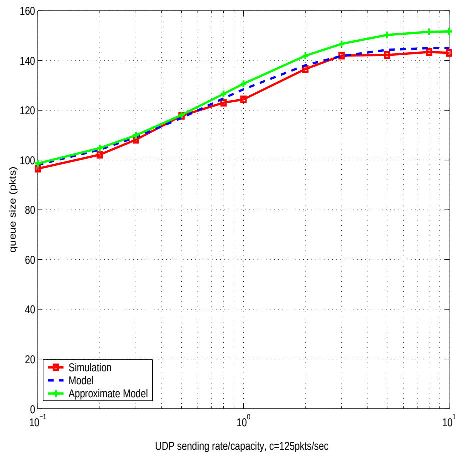
A. Experiment 1: macroscopic behavior

We vary UDP sending rate x_0 from 0.1Mbps to 10Mbps, corresponding to $0.1c$ to $10c$, and vary the number N of TCP flows from 12 to 64, to observe their effect on the equilibrium behavior of TCP/CHOKe. We measure the following quantities

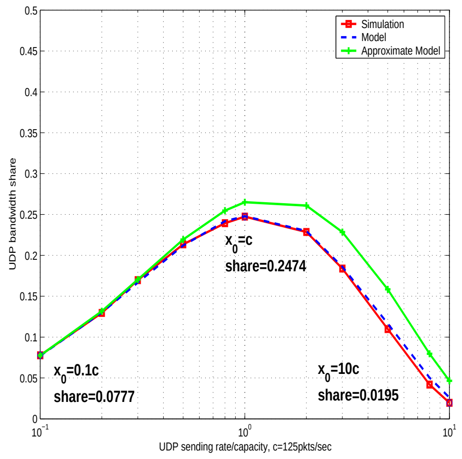
1) aggregate queue size b
2) UDP bandwidth share $\mu_0 = \rho_0(b)$
3) TCP throughput $x_1(1-p_1) = \mu_1 c = \rho_1(b)c$

as functions of x_0/c and of N. We then solve for these quantities using our analytical model (3)–(9), and the approximate model described in Section III-C. The results, shown in Figures 4 and 5, illustrate both the macroscopic behavior of TCP/CHOKe and the accuracy of our analytical models. We now discuss these results in detail.

First we study the effect of UDP sending rate on queue size and bandwidth allocation. The number of TCP sources is fixed at $N = 32$. As can be seen from Figure 4, the aggregate



(a) Queue size b



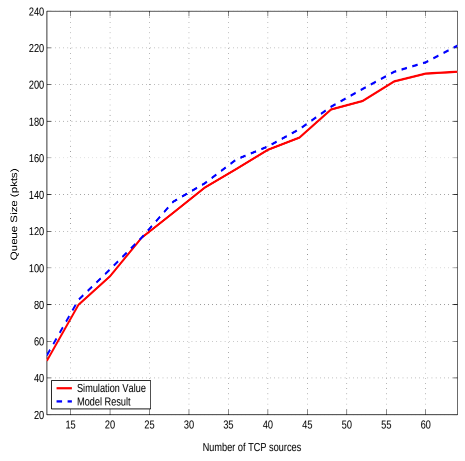
(b) UDP bandwidth share μ_0

Fig. 4. Experiment 1: Effect of UDP rate x_0 on queue size and UDP share. $N = 32$, $x_0 = 0.1c$ to $10c$, $c = 1$Mbps, simulation duration = 300sec.

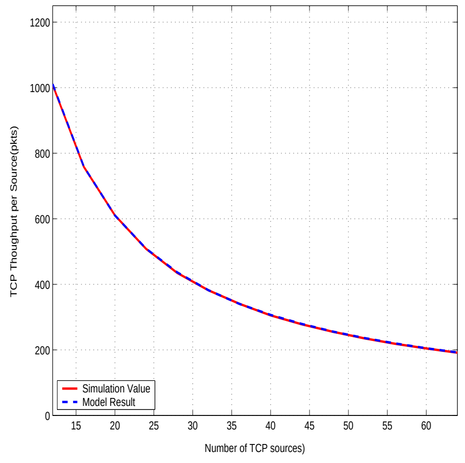
queue length b steadily increases as UDP rate x_0 rises. UDP bandwidth share $\mu_0 = \rho_0(b)$ rises, peaks, and then drops to less than 5% as x_0 increases from $0.1c$ to $10c$, while the total TCP throughput follows an opposite trend, eventually exceeding 95% of the capacity (not shown). These results match closely those obtained in [11], and with both the analytical model (3)–(9) and the approximate model of Section III-C.

Figure (4(b) also displays the UDP bandwidth share measured from the simulations for the cases $x_0 = 0.1c, c, 10c$. It verifies Theorems 1 and 2 which predict that the UDP bandwidth share peaks at around 0.269 and tends to zero as x_0 increases. Simulation and numerical solution using the full model (3)–(9) both show a smaller UDP share than that predicted by the approximate model (Theorem 1). This is because the theorem is derived under three approximations that require large N and large b. In Section V-C below, we will show the corresponding results for TCP Vegas, where N and b are both larger and the match between simulation and approximate model is better.

Next, we fix $c = 1$Mbps, $x_0 = 10c$, and vary N from 12 to 64. Figure 5 shows the effect of N on aggregate queue size b and on per-flow TCP throughput $\mu_1 c = \rho_1(b)c$. As expected, the queue



(a) Queue size b



(b) TCP throughput per flow $\mu_1 c$

Fig. 5. Experiment 1: Effect of number N of TCP flows on aggregate queue b and per-flow TCP throughput $\mu_1 c$. $N = 12 - 64$, $x_0 = 1250$ pkts/s, $c = 125$pkts/s, simulation duration = 300s

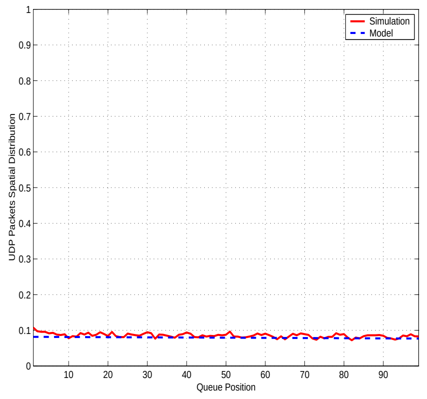
size increases and per-flow TCP throughput decreases with N as the queue becomes more congested. Again, the simulation and analytical results match very well, further validating our model.

B. *Experiment 2: spatial distribution* $\rho_0(y)$

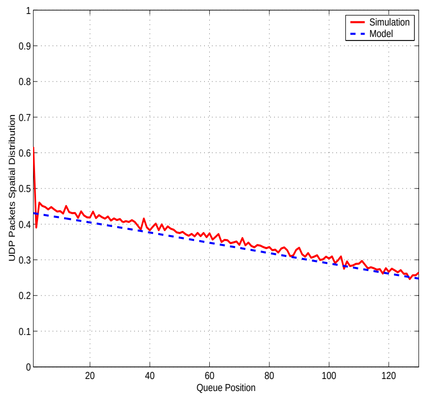
This set of results measure the spatial distributions $\rho_0(y)$ of UDP packets in the set of simulations shown in Figure 4, with parameters: $c = 1$Mbps, $N = 32$, x_0 varies from $0.1c$ to $100c$. The simulation results, and analytical solutions, are both shown in Figure 6. They match well Theorems 4 and 8; compare with Figure 2(b) in Section IV-C.

To measure the packet distribution from each simulation (x_0 value), we took $J = 3000$ snapshots of the queue every 100ms for 300 seconds. From the J sample queue sizes b^j, we first calculated the average $b_{avg} := \sum_j b^j / J$. The distribution was estimated over this range $[0, b_{avg}]$, as follows. For each $y \in [0, b_{avg}]$, the sample distribution is calculated as

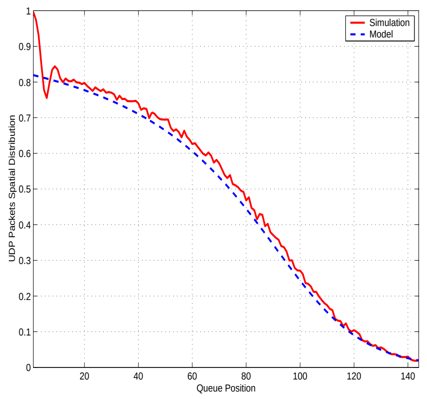
$$\rho(y) \quad = \quad \frac{1}{J} \sum_j \mathbf{1}_j(y)$$



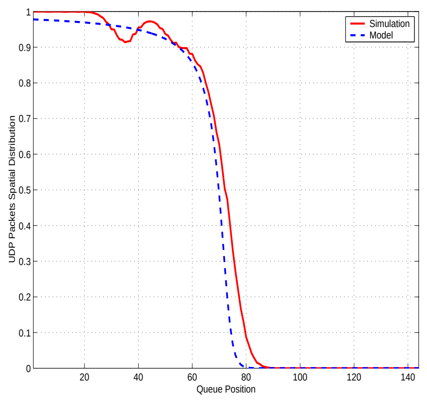
(a) UDP rate $x_0 = 0.1c$



(b) UDP rate $x_0 = c$



(c) UDP rate $x_0 = 10c$



(d) UDP rate $x_0 = 100c$

Fig. 6. Experiment 2: Spatial distribution $\rho(y)$ of packets in queue at different UDP rates x_0. $N = 32$, $c = 125$pkts/s, simulation duration = 300sec. Verifies Theorems 4 and 8 (compare with Figure 2(b)).

where $\mathbf{1}_j(y)$ is 1 if the packet in position $\lfloor yb^j/b_{avg} \rfloor$ of the jth snapshot is UDP, and 0 otherwise.

When $x_0 = 0.1c$ (Figure 6(a)), UDP packets are distributed roughly uniformly in the queue, with probability close to 0.08 at each position. As a result, its bandwidth share is roughly 10%. As x_0 increase, $\rho_0(y)$ concentrates more and more near the tail of the queue and drops rapidly toward the head, as predicted by Theorems 4 and 8.

Also marked in Figure 4(b), are the UDP bandwidth shares corresponding to UDP rates in Figure 6. As expected the UDP bandwidth shares in 4(b) are equal to $\rho_0(b)$ in Figure 6. When $x_0 > 10c$, even though roughly half of the queue is occupied by UDP packets, almost all of them are dropped before they reach the head of the queue!

C. *Experiment 3: Vegas*

In this subsection, we present similar simulations with TCP Vegas and compare with those with TCP NewReno. They illustrate that the qualitative behavior of TCP/CHOKe is insensitive to the specific TCP algorithms. It also shows that Vegas scales better than Reno with respect to link capacity (Vegas simulations used 15 times the link capacity in NewReno simulations), especially under CHOKe. This is because CHOKe increases the overall loss probability, limiting the achievable rate of TCP Reno (see also Section VI). Since TCP Vegas sets its rate based on queueing delay, it does not have this limitation. See [16] for more simulation results with Vegas.

For TCP Vegas, the source rate x_1 is related to the round trip time $d + \tau$ in equilibrium according to (see [6]):

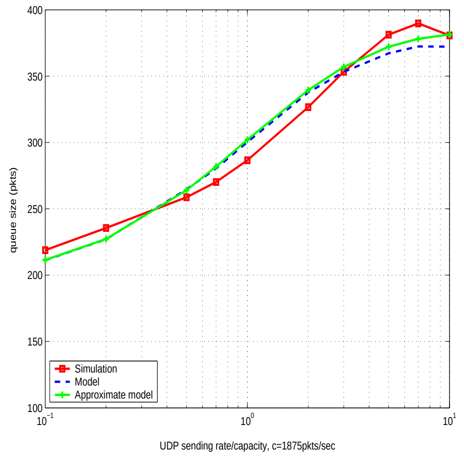
$$x_1 = \frac{\alpha d}{\tau}$$

where α is a protocol parameter. If the buffer is not leaky, each Vegas source puts αd number of packets in the queue and hence the total number of TCP packets in the queue is $N\alpha d$.

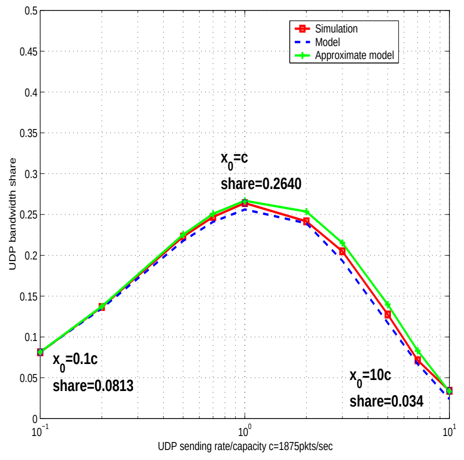
The original Vegas implementation in ns-2 works poorly in a lossy environment, for two reasons. First the implementation estimates RTT naively by setting it to the difference between sending time of a packet and receiving time of its ACK. When packets are lost, the ACK may be a duplicate ACK or it may be triggered by the retransmitted packet. A more sophisticated estimation is required when losses are frequent. Second, when there are multiple losses in the same round trip time, which is not infrequent since CHOKe drops two packets every time a comparison yields a match, the Vegas implementation often incurs timeout and slow-start. We re-implemented the Vegas module in ns-2 based on the TCP-NewReno code which better handles losses. There are three major changes. First, we do not estimate RTT with duplicate ACKs, especially with the first and second duplicate ACKs when fast retransmit/fast recover phase is not yet entered. Second we use the NewReno's fast retransmit and fast recovery code to deal with multiple losses. Third, we change the Vegas code such that it does not halve the sending window when there is a loss.

For all Vegas simulations, the link capacity is fixed at $c = 15$Mbps the round trip propagation delay is $d = 100$ms, the number of (identical) TCP Vegas flows is $N = 100$. We set $\alpha d = 20$ packets for all Vegas flows. We use RED+CHOKe as the queue management with RED parameters: (min_th $\underline{b} = 20$ packets, max_th $\overline{b} = 1,020$ packets, p_max = 0.1). Packet size is 1KBytes. Simulation time is 20 seconds.

We vary UDP sending rate x_0 from $0.1c = 1.5$Mbps to $10c = 150$Mbps. We measure the UDP bandwidth share $\mu_0 = x_0(1-p_0)/c$ and aggregate queue length b, and compare them with the numerical solutions of the full model (3)–(9) and those of the approximate model described in Section III-C. The results are shown in Figure 7. Comparison of this with Figure 4 for



(a) Queue size b



(b) UDP bandwidth share μ_0

Fig. 7. Experiment 3: Effect of UDP rate x_0 on queue size and UDP share. $N = 100$, $x_0 = 0.1c$ to $10c$, $c = 15$Mbps, simulation duration = 20sec.

NewReno simulations confirms that the qualitative behavior of TCP/CHOKe is insensitive to TCP algorithms.

Theorems 1 and 2 predict that the UDP bandwidth share peaks at around 0.269 and tends to zero as x_0 increases. Figure 7(b) also displays the UDP share from simulations with $x_0 = 0.1c, c, 10c$. As mentioned in Section V-A, both simulation and numerical solution of the full model (3)–(9) yield a smaller maximum UDP throughput than predicted, because of the three approximations used in deriving the theorems. Since the number of flows is larger in Vegas simulations ($N = 100$) than in NewReno simulations ($N = 32$), and the queue length b is larger in Vegas simulations than in NewReno simulations (compare Figures 7(a) with Figure 4(a)), the match between simulation and Theorem 1 is better for Vegas than for NewReno (compare Figure 7(b) with Figure 4(b)).

VI. Discussion

Our model captures well the equilibrium behavior of CHOKe under the assumption that the queue size b is between $\underline{b}$ and $\overline{b}$. This holds if c is sufficiently small or N is sufficiently large. A sample queue size from a NewReno simulation is shown in Figure 8, where indeed the queue size fluctuates around a level much bigger than $\underline{b}$ = 20pkts. This may not hold for small N.

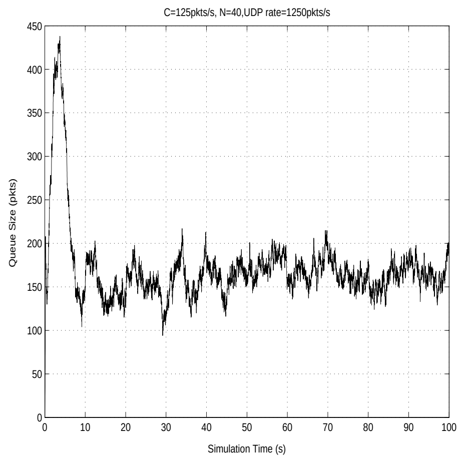


Fig. 8. Queue size with $N = 40$.

With smaller N, each TCP source gets a larger bandwidth share, which requires a lower dropping probability. However, when CHOKe is active, it imposes a lower bound on the dropping probability: from (1)

$$\begin{aligned} p_i &= 2h_i + (1-h_i)r \\ &\geq 2h_i \geq \frac{1}{N} \end{aligned} \tag{45}$$

where the last inequality follows from (5) and the fact that UDP packets occupy at most half of the queue (Theorem 5(1)).

We can estimate the minimum N with which CHOKe is always active. Approximate the TCP function in (43) by

$$p_1 = \frac{2}{x_1^2(d+\tau)^2}$$

Combining with (45) to get

$$\frac{2N}{x_1^2(d+\tau)^2} \geq 1$$

When UDP sending rate is large, TCP flows take almost all the bandwidth, so

$$x_1 \approx \frac{c}{N}$$

Around $\underline{b}$, queueing delay is roughly

$$\tau \approx \frac{\underline{b}}{c}$$

Putting all these together, the minimum number of TCP flows required for CHOKe to remain active is roughly

$$N \geq \sqrt[3]{\frac{(cd+\underline{b})^2}{2}}$$

Using the same parameters as in the last section, we can estimate the minimal N to be 8.08. When $N = 8$, queue size oscillates around $\underline{b}$ = 20 packets, constantly turning CHOKe on and off, as shown in Figure 9 (compare with Figure 8). When N is small, the equilibrium model in Section II no longer holds. The same phenomenon is observed when c increases (with fixed N). The lower bound on dropping probability when CHOKe is

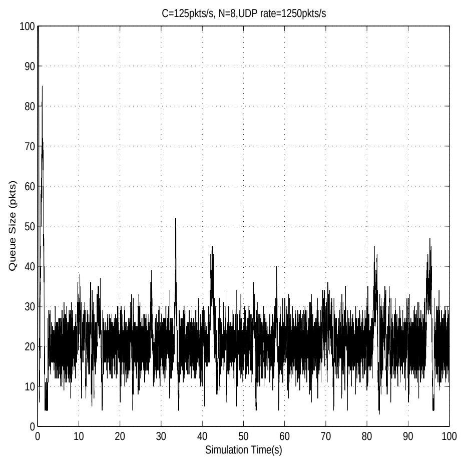


Fig. 9. Queue size with $N = 8$.

active, $p_1 \geq 1/N$, eventually prevents TCP flows from making full use of the available capacity. An interesting positive effect is that the queue length is controlled to stay around $\underline{b}$.

We have also simulated with more than one UDP flows, and with Back CHOKe (where an arrival is always compared with the packet at the tail of the queue) and Front CHOKe (where an arrival is always compared with the packet at head of the queue) [11]. When there is more than one UDP flow, CHOKe may not be as effective in protecting TCP traffic and UDP flows may take a larger bandwidth share at high sending rate. The throughput behavior of CHOKe variants, where burstiness has a much stronger effect, can be quite different from that of the original CHOKe. For instance, with Back CHOKe, once admitted into the buffer, a packet will not be dropped. When the UDP input rate is high, all packets in the same burst of UDP arrivals will be dropped, except possibly the last packet when the burst has odd number of packets. As a result the UDP share remains small (in fact close to 1/3 if the number N of TCP flows is large) and the queue oscillates around $\underline{b}$.

VII. Conclusions

CHOKe is completely stateless and extremely simple to implement, yet surprisingly effective in bounding the bandwidth share of UDP flows. As shown in the simulations of [11], as UDP source rate increases, its bandwidth share eventually drops to zero, exactly opposite to how a regular FIFO buffer behaves. To understand this precisely requires a careful study of the queueing process under CHOKe.

In this paper, we have developed a model of CHOKe. Its key features are the incorporation of the feedback equilibrium of TCP and a detailed modeling of the queue dynamics. We prove that as UDP input rate increases, its bandwidth peaks at $(e+1)^{-1} = 0.269$ when UDP input rate is slightly larger than link capacity, and drops to zero as UDP input rate tends to infinity. To explain this phenomenon, we have introduced the concepts of spatial distribution and velocity of packets in the queue. We prove structural and asymptotic properties of these quantities that make it possible for UDP to simultaneously maintain a large number of packets in the queue and receive a vanishingly small bandwidth share, the mechanism through which CHOKe protects TCP flows.

Finally, we remark that CHOKe algorithm may be constantly turned on and off when the link capacity is high or the number of TCP sources is small. This can prevent TCP flows from making full use of available capacity but regulate the queue size to around $\underline{b}$.

Our model applies only to the equilibrium behavior of TCP/CHOKe which presumes an asymptotically stable system. It is restricted to the simple case of homogeneous TCP flows, a single UDP flow, a single drop candidate, at a single bottleneck link. It would be interesting to extend the analysis to a more general setting. The technique presented here may also be applicable to analyzing other types of leaky buffer.

Acknowledgment: We are grateful to Rong Pan of Stanford for sharing the codes and Ke Wang of Caltech for useful discussions.

VIII. Appendix

A. *Proof of Theorem 3*

Using (27), (35) can be rewritten as

$$v'(y) \quad = \quad \beta(\rho_0(y)x_0 + N\rho_1(y)x_1) \quad < \quad 0$$

where $\beta = \ln(1 - 1/b) < 0$. Hence $v(y)$ is strictly decreasing.

Differentiating again and using $\rho_0'(y) + N\rho_1'(y) = 0$ from (27), we have

$$\begin{aligned} v''(y) &= \beta(\rho_0'(y)x_0 + x_1\rho_1'(y)N) \\ &= \beta(x_0 - x_1)\rho_0'(y) \end{aligned}$$

From (36), we have

$$v''(y) \quad = \quad \beta^2(x_0 - x_1)^2\rho_0(y)\rho_1(y)\frac{N}{v(y)} \quad \geq \quad 0$$

with equality if and only if $x_0 = x_1$. Hence $v(y)$ is convex and is linear if and only if $x_0 = x_1$.

The boundary values of $v(y)$ follows from (30) (sum over i) and (31). □

B. *Proof of Theorem 4*

From (36), we have

$$\rho_0'(y) \quad = \quad \beta(x_0 - x_1)\rho_0(y)(1 - \rho_0(y))\frac{1}{v(y)}$$

Since $\beta < 0$ and $x_0 \geq x_1$, $\rho_0'(y) \leq 0$, i.e., $\rho_0(y)$ is a decreasing function. The value of $\rho_0(0)$ follows from (30) and Theorem 3.

Differentiating (35), we have after some algebra

$$\rho_0''(y) = \frac{\beta^2(x_0 - x_1)\rho_0(y)(1 - \rho_0(y))}{v^2(y)} \cdot ((1 - 3\rho_0(y))(x_0 - x_1) - x_1)$$

There are three possible cases:

- $\rho_0(0) \leq \rho^*$: $(1 - 3\rho_0(y))(x_0 - x_1) - x_1 \geq 0$ and $\rho_0''(y) \geq 0$. In this case $\rho(y)$ is convex decreasing.
- $\rho_0(b) \geq \rho^*$: $(1 - 3\rho_0(y))(x_0 - x_1) - x_1 \leq 0$ and $\rho_0''(y) \leq 0$. In this case $\rho(y)$ is concave decreasing.
- $\rho_0(0) > y^* > \rho_0(b)$: Then $\rho_0''(y) \leq 0$ for $y \leq y^*$ and $\rho_0''(y) \geq 0$ for $y \geq y^*$. Hence $\rho(y)$ is strictly concave decreasing before the inflexion point y^* and strictly convex decreasing after.

□

C. Proof of Lemma 7

We will show that the numerator of (37) tends to 0 as $x_0 \to \infty$. From Theorem 4, we have

$$\rho_0(0) = \frac{x_0(1-h_0)}{x_0(1-h_0) + Nx_1(1-h_1)} \tag{46}$$

and hence from (38), we have

$$a = \frac{x_0(1-h_0)}{Nx_1(1-h_1)}$$

The numerator of (37) is then

$$\frac{x_0(1-h_0)}{Nx_1(1-h_1)} \cdot e^{\beta(x_0-x_1)\tau(y)}$$

From Theorem 5, as $x_0 \to \infty$, $h_i^\infty < 1$, $i = 0, 1$, and $x_1^\infty < \infty$. Moreover, from Theorem 5(d), $b^\infty < \infty$ and hence $\beta^\infty = \ln(1 - 1/b^\infty) < 0$. Then, if $\lim_{x_0} \tau(y) > 0$, it can be shown that a grows at most linearly to ∞, while $e^{\beta(x_0-x_1)\tau(y)}$ tends at least exponentially to 0. Hence the numerator $ae^{\beta(x_0-x_1)\tau} \to 0$. □

D. Proof of Theorem 8

From Theorem 6, we know $\rho_0^\infty(b) = 0$. Hence we can define

$$\hat{y} \;:=\; \inf\{\, y \mid \rho_0^\infty(y) = 0 \,\}$$

Let y' be any point with $\rho_0^\infty(y') = 1$. One exists because, from proof of Lemma 7, we know $\rho_0^\infty(0) = 1$ (see (46)). Consider the midpoint y'' between y' and $\hat{y}$, $y'' = (y' + \hat{y})/2$. It suffices to prove that (i) either $\rho_0^\infty(y'') = 1$ or $y'' = \hat{y}$, and (ii) $\hat{y} = y^*$.

(i) Suppose $\rho_0^\infty(y'') < 1$. We need to show that $y'' = \hat{y}$. Since $\rho_0(y)$ is decreasing y for any finite $x_0 \geq x_1$ (Theorem 4), its limit $\rho_0^\infty(y)$ is nonincreasing in y. Hence, $y'' \leq \hat{y}$. Suppose for the sake of contradiction that $\delta := \hat{y} - y'' > 0$.

We first prove that $v^\infty(y'') < \infty$. Consider for any finite $x_0 \geq 0$ the quantity $\rho_1(y)v(y)$, the TCP flow rate at position y'' in the queue. Using (32) and (30), we have

$$\rho_1(y'')v(y'') \;\leq\; \rho_1(0)v(0) \;=\; x_1(1-r)(1-h_1)$$

Taking limit as $x_0 \to \infty$, we have

$$\rho_1(y'')v(y'') \;\leq\; x_1^\infty(1-r^\infty)(1-h_1^\infty)$$

which is bounded by Theorem 5. Note that $\rho_0^\infty(y'') < 1$ by definition of y''. Since $\rho_0(y'') + N\rho_1(y'') = 1$, we have (taking limit of the corresponding finite-x_0 expression as $x_0 \to \infty$)

$$\rho_0^\infty(y'') \;=\; \frac{\rho_0^\infty(y'')v^\infty(y'')}{\rho_0^\infty(y'')v^\infty(y'') + N\rho_1^\infty(y'')v^\infty(y'')} \;<\; 1$$

Hence, $\rho_1^\infty(y'')v^\infty(y'') < \infty$ implies $\rho_0^\infty(y'')v^\infty(y'') < \infty$. This in turn implies that $v^\infty(y'') < \infty$ since $\rho_0^\infty(y'') > 0$ by definition of y'' (otherwise, $y'' = \hat{y}$ and we are done).

Now define y''' be the midpoint of y'' and $\hat{y}$, $y''' = (y'' + \hat{y})/2$. Then $\rho_0^\infty(y''') > 0$ by definition of $\hat{y}$. We now show that $v^\infty(y'') < \infty$ implies $\rho_0^\infty(y''') = 0$, a contradiction.

From Theorem 3, $v(y)$ is strictly decreasing in y and hence $\tau(y)$ defined by (26) satisfies

$$\begin{aligned}
\tau(y''') &= \int_0^{y'''} \frac{ds}{v(s)} \\
&\geq \int_{y''}^{y'''} \frac{ds}{v(s)} \\
&> \int_{y''}^{y'''} \frac{ds}{v(y'')} \\
&= \frac{\delta}{2v(y'')}
\end{aligned}$$

Taking limit as $x_0 \to \infty$, we have

$$\tau^\infty(y''') \;\geq\; \frac{\delta}{2v^\infty(y'')} \;>\; 0$$

where the last inequality follows from $v^\infty(y'') < \infty$. Hence $\rho_0^\infty(y''') = 0$ by Lemma 7. But this contradicts the definition of $\hat{y}$ since $y''' < \hat{y}$. Hence we must have $y'' = \hat{y}$.

(ii) The above shows that $\rho_0^\infty(y)$ takes the form shown in Figure 2. Then, by (28) and Theorem 5, we have (using assumption A4)

$$\frac{1-r^\infty}{2-r^\infty}\, b^\infty \;=\; \int_0^{\hat{y}} dy \;=\; \hat{y}$$

This completes the proof. □

E. Proof of Theorem 9

We first prove for $y < y^*$ and then for $y > y^*$.

Assume there exists $y' < y^*$ such that $v^\infty(y') < \infty$. Consider $y'' = (y' + y^*)/2$. Since $v^\infty(y)$ is nonincreasing in y, we have $v^\infty(y) \leq v^\infty(y') < \infty$ for any $y' \leq y \leq y''$. Hence, using assumption A4, we have

$$\begin{aligned}
\tau^\infty(y'') &= \int_0^{y''} \frac{ds}{v^\infty(s)} \\
&\geq \int_{y'}^{y''} \frac{ds}{v^\infty(s)} \\
&\geq \frac{y''-y'}{v^\infty(y')} \;>\; 0
\end{aligned}$$

Then Lemma 7 implies that $\rho_0^\infty(y'') = 0$. Since $y'' < y^*$, this contradicts theorem 8. So for $y < y^*$, $v^\infty(y) = \infty$.

For $y > y^*$, we prove the theorem in 4 steps.

Step 1: $v^\infty(y) < \infty$ for all $y > y^*$.

Using (32) and (30), and taking limit as $x_0 \to \infty$, we have

$$\rho_1^\infty(y)v^\infty(y) \;\leq\; \rho_1^\infty(0)v^\infty(0) \;=\; x_1^\infty(1-r^\infty)(1-h_1^\infty)$$

But by Theorem 8, $\rho_1^\infty(y) = (1-\rho_0^\infty(y))/N = 1/N$ for $y > y^*$, and by Theorem 5, the right-hand side is finite. Hence $v^\infty(y) < \infty$ for all $y > y^*$.

Step 2: $\tau^\infty(y) > 0$ for $y > y^*$.

Fix $y > y^*$. Since $v(y)$ is strictly decreasing (Theorem 3) we have, for each finite $x_0 \geq 0$,

$$\begin{aligned}
\tau(y) &= \int_0^{y} \frac{ds}{v(s)} \\
&> \int_{\frac{y^*+y}{2}}^{y} \frac{ds}{v(s)} \\
&> \frac{y-y^*}{2v\left(\frac{y^*+y}{2}\right)}
\end{aligned}$$

Taking limit as $x_0 \to \infty$, we have for $y > y^*$

$$\tau^\infty(y) \quad \geq \quad \frac{y - y^*}{2v^\infty\left(\frac{y^*+y}{2}\right)} \ > \ 0$$

Step 3: There exists an integrable function $\overline{H}^{x_0}(y)$ such that, for all $x_0 \geq 0$ and $y > y^*$,

$$H^{x_0}(y) \quad := \quad x_0\rho_0(y) + x_1(1-\rho_0(y)) \ \leq \ \overline{H}^{x_0}(y)$$

From (37) we have

$$x_0\rho_0(y) \quad = \quad \frac{x_0 a e^{\beta(x_0-x_1)\tau(y)}}{1 + a e^{\beta(x_0-x_1)\tau(y)}}$$

Since $\tau^\infty(y) > 0$ for all $y > y^*$, the proof of Lemma 7 shows that, as $x_0 \to \infty$, a grows at most logarithmically to ∞ while $e^{\beta(x_0-x_1)\tau(y)}$ grows at least exponentially to 0. Hence the numerator $x_0 a e^{\beta(x_0-x_1)\tau(y)}$ tends to 0 as $x_0 \to \infty$. Moreover,

$$m(y) \quad := \quad \max_{x_0 \geq 0} \ x_0 a e^{\beta(x_0-x_1)\tau(y)}$$

is finite. Hence, for x_0 sufficiently large (so that $a > 0$), we have

$$x_0\rho_0(y) \quad \leq \quad m(y)$$

Note that $m(y)$ is independent of x_0. Hence if we define $\overline{H}^{x_0}(y) = m(y) + x_1$, then $H^{x_0}(y) \leq \overline{H}^{x_0}(y)$. Since x_1 is finite for all x_0 (Theorem 5), $\overline{H}^{x_0}(y)$ is integrable over (y^*, b).[3]

Step 4: $v^\infty(y) = c - \beta^\infty x_1^\infty \ (b^\infty - y)$ for $y > y^*$.

Fixed $y > y^*$. From (35), we have, for each $x_0 \geq 0$,

$$\begin{aligned} v(y) \quad &= \quad c - \beta \int_y^b (\rho_0(s)x_0 + (1-\rho_0(s))x_1) ds \\ &= \quad c - \beta \int_y^b H^{x_0}(s) ds \end{aligned}$$

Taking limit as $x_0 \to \infty$, we have

$$v^\infty(y) \quad = \quad c - \beta \lim_{x_0 \to \infty} \int_y^b H^{x_0}(s) ds$$

From Step 3, $H^{x_0}(s)$ is upper bounded by an integrable function and converges pointwise to $H^\infty(y) = x_1^\infty$ as $x_0 \to \infty$. Hence Lebesgue convergence theorem applies (see [13, p.229], [1, pp. 215]):

$$\lim_{x_0 \to \infty} \int_y^b H^{x_0}(s) ds \quad = \quad \int_y^{b^\infty} H^\infty(s) ds \quad = \quad x_1^\infty \ (b^\infty - y)$$

Hence, $v^\infty(y) = c - \beta^\infty x_1^\infty \ (b^\infty - y)$ for $y > y^*$.

This completes the proof. □

REFERENCES

[1] Patrick Billingsley. *Probability and Measure*. John Wiley & Sons, 2 edition, 1986.

[2] W. Feng, K G. Shin, D. Kandlur, and D. Saha. Stochastic Fair Blue: A queue management algorithm for enforcing fairness. In *Proceedings of INFOCOM*, April 2001.

[3] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, 1(4):397–413, August 1993. `ftp://ftp.ee.lbl.gov/papers/early.ps.gz`.

[4] Dong Lin and Robert Morris. Dynamics of random early detection. In *Proceedings of SIGCOMM'97*, pages 127–137, September 1997. `http://www.acm.org/sigcomm/sigcomm97/papers/p078.ps`.

[5] Steven H. Low. A duality model of TCP and queue management algorithms. *IEEE/ACM Trans. on Networking*, 11(4):525–536, August 2003. `http://netlab.caltech.edu`.

[6] Steven H. Low, Larry Peterson, and Limin Wang. Understanding Vegas: a duality model. *J. of ACM*, 49(2):207–235, March 2002. `http://netlab.caltech.edu`.

[7] P. McKenny. Stochastic fairness queueing. In *Proceedings of Infocom*, pages 733–740, 1990.

[8] J. A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, pages 308–313, 1965.

[9] T. J. Ott, T. V. Lakshman, and L. Wong. SRED: Stabilized RED. In *Proceedings of IEEE Infocom'99*, March 1999. `ftp://ftp.bellcore.com/pub/tjo/SRED.ps`.

[10] Rong Pan, Chandra Nair, Brian Yang, and Balaji Prabhakar. Packet dropping mechanisms: some examples and analysis. In *Proc. of 38th Annual Alleen Conference on Communication, Control, and Computing*, October 2001.

[11] Rong Pan, Balaji Prabhakar, and Konstantinos Psounis. CHOKe: a stateless AQM scheme for approximating fair bandwidth allocation. In *Proceedings of IEEE Infocom*, March 2000.

[12] Lawrence Perko. *Differential equations and dynamical systems*. Springer, 3 edition, 2001.

[13] H. L. Royden. *Real analysis*. MacMillan, 1968.

[14] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks. In *Proceedings of ACM Sigcomm*, 1998.

[15] Ao Tang, Jiantao Wang, and Steven H. Low. Understanding CHOKe. In *Proc. of IEEE Infocom*, April 2003. `http://netlab.caltech.edu`.

[16] Jiantao Wang, Ao Tang, and Steven H. Low. Maximum and asymptotic UDP throughput under CHOKe. In *Proc. of ACM Sigmetrics*, June 2003

[3]To be precise, $b = b^{x_0}$ is a function of x_0. To make the domain of integration independent of x_0, extend $H^{x_0}(y)$ and $\overline{H}^{x_0}(y)$ to $(y^*, \max_{x_0} b^{x_0})$ by defining them to be zero for y not in (y^*, b^{x_0}).