

# Enhancing Congestion Control for Wireless Links

Saâd Biaz

Dai Yawen

Computer Science and Software Engineering Department

Auburn University

Auburn, AL 36849-5347, USA

{sbiaz,dyeaiya}@eng.auburn.edu

**Keywords:** Basestation congestion control, TCP performance, wireless networks

## Abstract

Bandwidth on wireless links keeps increasing, so does the bandwidth on wired links. Therefore, the discrepancy in bandwidth between wireless and wired links still persists. Such discrepancy causes the wireless link to likely remain the bottleneck for data transfers from wired networks to a wireless destination,

As a bottleneck, the basestation may experience congestion losses that can adversely impact TCP performance. We propose a technique that substantially alleviates the congestion at the base station and enforces fairness among the TCP connections that are sharing the wireless link. Our technique does not require any change at the TCP sender or the TCP receiver. We evaluate our technique using simulations with ns-2. Preliminary results yield almost a null congestion loss rate, a decrease in average queue length, and an increase in the throughput. Fairness is very well enforced.

## 1 Introduction

Widespread use of wireless networks is becoming a reality. Many wireless Internet Service Providers such as GoAmerica, Wayport, or SurfnSip are already competing for this tremendous market [5]. In December 2002, AT&T, Intel, and IBM created Cometa Networks, a group that will provide wireless Internet access nationwide. The wireless access points will certainly be using 802.11b which offers a bandwidth up to 11 Mbps, a tremendous bandwidth for wireless links. However, wired links are also offering increasing bandwidths: in June 2002, the IEEE Standards Board approved the standard IEEE Std 802.3ae for a 10Gb/s Ethernet. Since backbone links are overprovisioned such that congestion losses in core routers are rare [4], the discrepancy in bandwidth elects the basestation most likely to

be the bottleneck with combined congestion and wireless losses, leading to a very poor TCP performance.

We propose a technique that we call Basestation Flow Control (**BFC**) that can alleviate congestion for TCP traffic at the basestation. **BFC** does not require any change at the TCP sender or the TCP receiver. The key of this technique is to tamper the window size field in the TCP header of all acknowledgments. The window size TCP header field is normally used by the receiver for flow control. In this work, we propose to extend the usage of this field to protect both the basestation and the ultimate receiver from overflowing. Such approach is justified by the special place of the basestation in data transfers from wired networks to wireless devices. We consider the wireless system (basestation+mobile) as an end-system. This is why we consider our scheme as a *flow control* mechanism, rather than a *congestion control* mechanism.

This paper is organized as follows: Section 2 introduces the reader to the network model and the problem. Related work is presented in Section 3. We present and describe our technique in Section 4. This technique is evaluated through simulation using ns-2. Section 5 describes our experiments. The results of these experiments are presented and discussed in Section 6. Section 7 is dedicated to our conclusion and future work.

## 2 The Problem

Figure 1 shows the network model that we consider. Mobile nodes are accessing the Internet through wireless points of attachment. These points of attachments are basestations with wired access to the Internet and wireless access to mobile nodes. Data transfers are mainly from servers to mobile nodes. In this work, our objective is to improve these data transfers that are destined to mobile nodes.

Due to the discrepancy in bandwidth between wired and wireless links, the basestation is likely to be the bottleneck for most transfers destined to mobile nodes. Moreover,

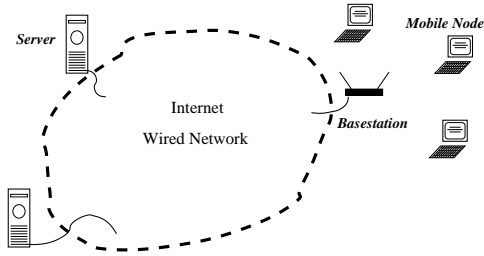


Figure 1: Network Model

bad radio conditions on the wireless link may adversely decrease the link layer throughput, leading to packets delayed at the basestation. If packets are delayed at the basestation, queue will build up. Such conditions lead to a situation where packets are lost due wireless losses and to congestion at the basestation. Surges in the contention for the wireless medium may also lead to the same situation. In summary, the basestation is likely to be the bottleneck for data transfers destined to a mobile node due to:

1. Discrepancy between wired and wireless bandwidth
2. High error loss rates.
3. High contention for the medium .

The idea is to consider the basestation and the mobile as one end-system that the sender should not overflow. To achieve this, we need two functions. **First**, we must estimate the optimal congestion window size that every TCP sender should use. **Second**, we have to “force” the TCP sender to adopt this optimal congestion window size. We present in the Section 4 how we achieve these two functions.

### 3 Related Work

The most relevant work is the *FDA* [6] flow control scheme for basestations. *FDA* works in conjunction with *Snoop* [1]. *Snoop* attempts to recover wireless losses without involving the sender. However, *Snoop* has to buffer packets sent on the wireless link and not yet acknowledged. If the wireless link experiences severe losses, *Snoop* may drown in severe buffer space problem. *FDA* proposes to alleviate the problem of buffer space by taking two actions: *FDA* forces a selected set of TCP senders to decrease their sending rates and drops optimistically from the buffer some packets not yet acknowledged.

Katabi proposes in [7] XCP, the eXplicit Control Protocol for high bandwidth-delay product networks. In XCP, the bottleneck router knows the current average round trip time (*rtt*) and congestion window size (*cwnd*) used by all XCP connections going through it. With the knowledge

of all *rtt*'s and *cwnd*'s, XCP implements two controllers: one for efficiency and one for fairness. The efficiency controller aims to keep the link busy while minimizing congestion losses. The fairness controller aims to enforce fairness among the XCP connections that are sharing the link. XCP is efficient, robust and stable. However, XCP requires modifications at the sender, at the receiver, and the intermediary nodes.

Our work is original in that it proposes a control algorithm that is more intuitive than XCP and does not require the knowledge of the current congestion window. We develop this control algorithm such that it can immediately be deployed without any change to TCP senders or TCP receivers.

## 4 Design and Implementation

In this section, we will first introduce the notation and terminology that we will use, and then will develop the **BFC** technique. We will develop this technique in two steps.

**First**, we will assume that *explicit information* is available at the basestation. We mean by *explicit information* the knowledge of the current average round trip time and current congestion window size for all TCP connections that share the wireless link. With such information, we compute the *optimal* congestion window size for each TCP connection based on its current round trip time. The use of the *optimal* congestion window by all TCP senders size aims to maximize the wireless link utilization, minimize congestion losses at the basestation, and enforce fairness. We call this technique *eXplicit BFC (XBFC)* because TCP senders and TCP receivers provide *explicit information* to the basestation to determine the optimal congestion window size.

**Second**, we develop a technique that does not require *explicit information*. Therefore, TCP senders or TCP receivers do not have to cooperate with the basestation. Our objective is to implement **BFC** to work with current TCP implementations. We call this technique *Blind BFC (BBFC)* to stress the fact that our technique does not get any assistance from TCP senders or TCP receivers.

### 4.1 Notations and Terminology

At the basestation, time is divided in epochs  $E^j$  of variable duration  $rtt^j$  ( $j \in \{0..\}$ ). We consider that we have  $n^j$  TCP senders  $S_i^j$  ( $i \in \{1, \dots, n^j\}$ ) that are sharing the bottleneck (the basestation) during epoch  $E^j$ . During epoch  $E^j$ ,  $N^j$  packets  $P_k^j$  ( $k \in \{1, \dots, N^j\}$ ) are sent by all TCP senders  $S_i^j$  and received by the basestation. In each packet  $P_k^j$ , TCP senders insert their current average round trip time  $rtt_i^j$  and current congestion window size  $cwnd_i^j$ . Based on this information, the basestation computes the *optimal* congestion

window size  $cwnd_{opt}^j(rtt_i^j)$  that sender  $S_i^j$  must use during the  $j^{th}$  epoch  $E^j$ . The value of  $cwnd_{opt}^j(rtt_i^j)$  is inserted on the packet  $P_k^j$ . When the TCP receiver gets packet  $P_k^j$  with the information  $cwnd_{opt}^j(rtt_i^j)$ , it will echo this value to the TCP sender in the acknowledgment.

- Normalized Epoch Window size  $NEW^j$ :  $NEW^j$  is the maximal number of packets that every sender  $S_i^j$  is allowed to send during epoch  $E^j$ . For fairness,  $NEW^j$  is the same for all TCP senders.
- Global average round trip time  $rtt^j$ :  $rtt^j$  is the estimate of the global (for all TCP connections) average round trip time made at the beginning of epoch  $E^j$ .  $rtt^j$  is estimated based on the round trip times  $rtt_k^{j-1}$  reported by packets  $P_k^{j-1}$  ( $k \in \{1, \dots, N^{j-1}\}$ ) during epoch  $E^{j-1}$ .  $rtt^j$  is defined as  $rtt^j = \frac{1}{N^{j-1}} \sum_{k=1}^{N^{j-1}} rtt_k^{j-1}$ .
- Global average congestion window size  $cwnd^j$ :  $cwnd^j$  is the global average congestion window defined as  $\frac{1}{N^{j-1}} \sum_{k=1}^{N^{j-1}} cwnd_k^{j-1}$ .  $cwnd^j$  is defined similarly to  $rtt^j$ .
- Wireless link bandwidth  $Bw$ :  $Bw$  is the bandwidth of the wireless link between the basestation and the mobile nodes.
- Estimated input rate  $\lambda^j$  to the basestation:  $\lambda^j$  is the estimate of the input data rate to the base station made at the beginning of epoch  $E^j$ .
- Average queue length  $Q^j$ :  $Q^j$  is the average queue length at the basestation at the beginning of epoch  $E^j$ .
- Maximum queue size  $Q_{Max}$ :  $Q_{Max}$  is the maximum queue size at the basestation.

## 4.2 eXplicit BFC (XBFC)

During the epoch  $E^j$  of duration  $rtt^j$ , a basestation can send up to  $Bw \times rtt^j$  packets on the wireless link. The basestation has to send  $Q^j$  packets (average backlog at the basestation) and the new packets sent by all TCP senders during epoch  $E^j$ . Then, all senders should not send more than  $(Bw \times rtt^j - Q^j)$  packets during epoch  $E^j$  if we want to eliminate the queue at the basestation. To enforce fairness, each sender must be able to send the same amount of data, i.e., the normalized epoch window size  $NEW^j$ . Suppose that there are  $n^j$  TCP senders  $S_i^j$  during epoch  $E^j$ . Then, each sender  $S_i^j$  should be allowed to send up to  $NEW^j = \lceil \frac{1}{n^j} (Bw \times rtt^j - Q^j) \rceil$  during epoch  $E^j$ .

Now, we have to determine the optimal congestion window size  $cwnd_{opt}^j$  that a TCP sender should use. The optimal congestion window size obviously depends on the current round trip time experienced by the TCP sender. As an example, suppose that the duration of epoch  $E^j$  is  $100ms$ , the round trip time  $rtt_i^j$  reported on packet  $P_k^j$  is  $20ms$ , and the normalized epoch window size  $NEW^j$  is 60 packets. During epoch  $E^j$ , the TCP sender with round trip time  $20ms$  can send 5 ( $\frac{100ms}{20ms}$ ) “windows of data”. Since the TCP sender can send up to  $NEW^j = 60$  packets,  $cwnd_{opt}^j(20ms)$  should be set to  $\frac{60}{\frac{100ms}{20ms}} = 12$ . For any packet  $P_k^j$  reporting a round trip time of  $rtt_k^j$ , we must set the optimal congestion window size to  $cwnd_{opt}^j(rtt_k^j) = \frac{NEW^j}{\frac{rtt^j}{rtt_k^j}}$ . We can summarize our formula as:

$$cwnd_{opt}^j(rtt_k^j) = \frac{\frac{1}{n^j} (Bw \times rtt^j - Q^j)}{\frac{rtt^j}{rtt_k^j}} \quad (1)$$

In order to compute the optimal congestion window  $cwnd_{opt}^j(rtt_k^j)$  size for each sender, we need at the basestation the current average round trip time for each TCP connection and the total number  $n^j$  of TCP connections per epoch  $E^j$ .

**Approximating  $n^j$ :** When the system reaches a steady state where all TCP connections are sending the same number of packets  $NEW^j$  during epoch  $E^j$ , we can trivially show that the average congestion window size  $cwnd^j$  is equal to the normalized epoch congestion window size  $NEW^j$ . Therefore, we can write that the estimated rate  $\lambda^j$  to the basestation is equal to  $n^j \times \frac{cwnd^j}{rtt^j}$ . Replacing  $n^j$ , Equation (1) becomes:

$$cwnd_{opt}^j(rtt_k^j) = \frac{\frac{cwnd^j}{\lambda^j} (Bw - \frac{Q^j}{rtt^j})}{\frac{rtt^j}{rtt_k^j}} \quad (2)$$

Due to the uncertainty on the estimate of the input rate  $\lambda^j$  and the estimate of the average queue size  $Q^j$ , we introduce three coefficients  $\alpha$ ,  $\beta$ , and  $\gamma$ . Moreover, the system does not reach the steady equilibrium as TCP connections are dynamically opened while others are closed. This may lead to temporary queue build up. In order to take into account the queue build up, we limit the optimal congestion window size by applying <sup>1</sup> a reducing factor  $(1 - \frac{Q^j}{Q_{Max}})$ .

In [2], we derive from Equation (2) the following expres-

<sup>1</sup> We apply the reducing factor twice when  $\frac{Q^j}{Q_{Max}} > 0.3$ .

sion for  $cwnd_{opt}^j(rtt_k^j)$

$$\gamma \left(1 - \frac{Q^j}{Q_{Max}}\right) \frac{cwnd^j \left(1 - \alpha + \frac{1}{\lambda^j} (\alpha Bw - \beta \frac{Q^j}{rtt^j})\right)}{\frac{rtt^j}{rtt_k^j}} \quad (3)$$

The coefficients  $\alpha$  and  $\beta$  are chosen with similar values as for XCP. In [7], these values ( $\alpha = 0.6$  and  $\beta = 0.226$ ) are shown to lead to a stable controller.  $\gamma$  is a conservative factor. The value  $\gamma = 0.85$  gave the best results. Further work must be done to justify these factors.

Our objective in the next section, is to develop *Blind BFC BBFC* such that no information is explicitly provided to the basestation. **BBFC** can be immediately deployed to work with current TCP implementations.

### 4.3 BBFC

The problem with **XBFC** is that it requires explicit cooperation from the TCP sender and the TCP receiver. Our objective is to develop **BBFC** to work with current TCP implementations. Therefore, no special cooperation from the TCP sender or TCP receiver should be sought. We have three problems to solve. First, we must estimate the round trip time  $t_r$  for every TCP connection. Second, we must estimate the current congestion window size  $cwnd$  for every TCP connection. Third, we must feedback TCP sender with the optimal congestion window size  $cwnd_{opt}^j(t_r)$ .

**Estimating Round Trip Times:** Measurements of round trip times for TCP connections may be done at the basestation using SYN packets as in [8]. However, SYN packets may experience different delays than data packets. This may be due to ARP delay (first packets) and special handling of SYN packets by routers. Another method is to get the measurements during the slow start phase of TCP. Figure 2 shows the timeline for one TCP connection. For the very first packets of a TCP connection, the largest difference in arrival time at the basestation of two packets from the same TCP connection is a good estimate of the round trip time. After slow start, packets get equally spaced based on the bandwidth at the bottleneck. Therefore, the round trip time for a connection may be measured well at the very beginning of the connection. It is possible to capture with good accuracy the large gaps  $rtt_0$  and  $rtt_1$  for every TCP connection. The average  $\frac{rtt_0 + rtt_1}{2}$  can be used as the basis of the estimation of the round trip time during the TCP connection lifetime. Round trip times and state variables used to make the measurements are kept in a hash table. Now, we must estimate the current congestion window size for each TCP connection.

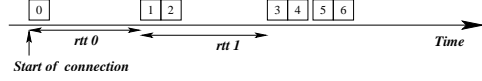


Figure 2: Packet Timing for One TCP Connection

**Estimating Congestion Window Size:** We maintain counters of the numbers of packets  $N_i^j$  sent by sender  $S_i^j$  and received by the basestation during epoch  $E^j$ . We can estimate the congestion window size  $cwnd_i^j$  used by sender  $S_i^j$  as  $\frac{N_i^j}{\frac{rtt^j}{rtt_i^j}}$  where  $rtt_i^j$  is the estimate of the round trip time for sender  $S_i^j$ . The estimate of the current congestion window size of each TCP connection is made at the beginning of each epoch  $E^j$ . Now, the remaining problem is to feedback the TCP senders with the optimal congestion window size  $cwnd_{opt}^j$  and “force” them to respect it. We examine this problem in the next subsection.

**Feedback to TCP connections:** Current TCP implementations do not include any mechanism that would allow a router to *advise* a TCP sender to use a given congestion window size. However, we can use the window size field of the TCP header to set an upper limit on the congestion window size. Whenever the basestation receives an acknowledgement destined to sender  $S_i^j$  with a window size field  $win$ , we search the hash table to get the optimal congestion window size  $cwnd_{opt}^j(i)$  corresponding to sender  $S_i^j$  for the current epoch  $E^j$ . The window size field in the acknowledgment is tampered and set to the value  $\min(win, cwnd_{opt}^j(i))$ . The limitations of using the window size field is that a TCP sender  $S_i^j$  with a current too small congestion window size will not take quickly advantage of the offered optimal congestion window size  $cwnd_{opt}^j(i)$ . We must observe that this feedback only sets up an upper limit on the congestion window size. The sender still uses the normal mechanism of TCP to reach this upper limit. In the following, we will present our experiments and results.

## 5 Experiments

We evaluate the eXplicit *BFC* (**XBFC1** and **XBFC2**) and **BBFC** schemes using simulations on  $ns-2$ . Figure 3 shows the simple topology used.

We have  $n$  senders  $S_i$  and one receiver. The (wired) links from the senders to the basestation have a bandwidth of 100 Mbps and a propagation delay of  $\delta_i$ . Variable values for  $\delta_i$  will allow us to simulate different round trip times. The wireless link has a bandwidth of 2 Mbps and a propagation delay of  $\delta_0$ . A variable propagation delay allows us to

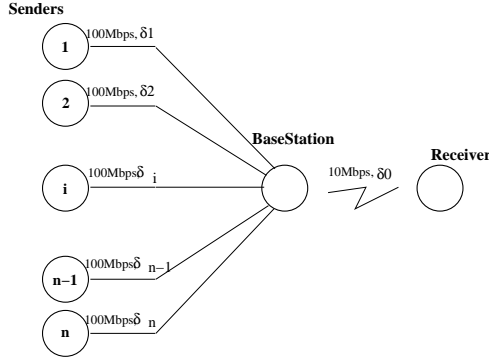


Figure 3: Topology of the Network

simulate satellite networks as well as wireless LANs. Experiments run long lived TCP connections up to 3000s in order to smooth out the effect of slow start.. For some set of experiments, we keep the number of TCP connections fixed and vary the round trip time of the TCP connections. For other experiments, we keep the round trip time fixed, and vary the number of TCP connections. The queue size is set to the bandwidth delay product. For experiments related to the fairness index, TCP connections have different round trip times. When the round trip times are equal for all TCP connections, it is known that TCP is quite fair.

## 6 Results and Discussion

We are mainly interested in the following metrics:

- Fairness index for the throughput as defined by Jain in [3]. Let  $tg_i$  be the throughput on TCP sender  $S_i$  ( $i \in \{1, \dots, n\}$ ). The fairness index is defined as  $\frac{(\sum_1^n tg_i)^2}{n \cdot \sum_1^n tg_i^2}$ . If all TCP connections get exactly equal shares in throughput then the fairness index is 1. Otherwise, it is smaller.
- Average bottleneck queue length  $Q$
- Average throughput of all connections.
- Wireless link utilization
- Packet drop probability

All the figures in this section present four curves. The four curves correspond respectively to TCP without basestation flow control (**TCP Alone**), TCP with eXplicit *BFC* (**XBFC1**) when all information is available, TCP with eXplicit *BFC* (**XBFC2**) when all information is available except the number of TCP connections and TCP with Blind *BFC* (**BBFC**).

### 6.1 Fairness Index

Figure 4 reports the fairness index on the  $y$ -axis. The  $x$ -axis

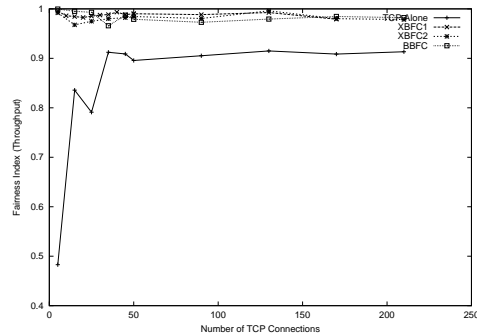


Figure 4: Jain and Chui's Index of Fairness

represents the number of TCP connections. For each point, we run one experiment with  $x$  long lived TCP connections. The one way propagation delay for the  $i^{th}$  TCP connection is equal to  $(50 + i \times 2)ms$  with  $i \in \{1, \dots, x\}$ . At the end of the experiment, we measure the throughput of each TCP connection and compute the fairness index as defined above.

First, we note that eXplicit **XBFC1** (**XBFC2**) and **BBFC** fairness indexes are almost indistinguishable and very close to 1. The explicit information available to **XBFC1** does not provide any advantage over **BBFC**.

Second, **TCP Alone** starts offering a good fairness index (higher than 0.90) as soon as 50 or more TCP connections share the basestation. This fairness is due to the fact that as the number of TCP connections increases, the congestion increases at the basestation. When congestion is heavy, it is unlikely that some subset of TCP connections would be able to develop a high throughput.

### 6.2 Bottleneck Average Queue Length

Figures 5 and 6 both show the average queue length on the  $y$ -axis. On Figure 5, the  $x$ -axis represents round trip times  $RTT$ . Each point corresponds to one experiment. In each experiment, 50 long lived TCP connections with the same round trip time  $RTT$  share the bottleneck. At the end of the experiments, we report the average queue length experienced by all TCP connections. First, we note that simple **XBFC1**, **XBFC2** and **BBFC** are again almost indistinguishable. **XBFC1** has a slightly smaller average queue size when the round trip time exceeds 1200ms.

Second, **TCP Alone** incurs a much higher average queue length than **XBFC1**, **XBFC2**, or **BBFC**. Average queue length increases with rtt as there is more outstanding data put on the network to achieve a given throughput.

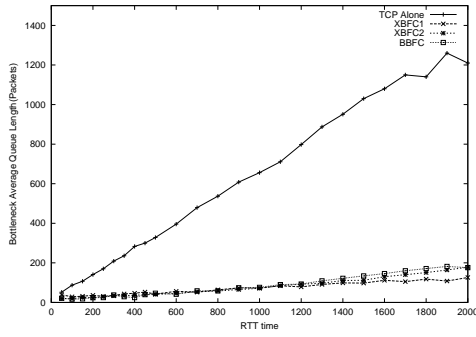


Figure 5: Average Queue Length vs RTT

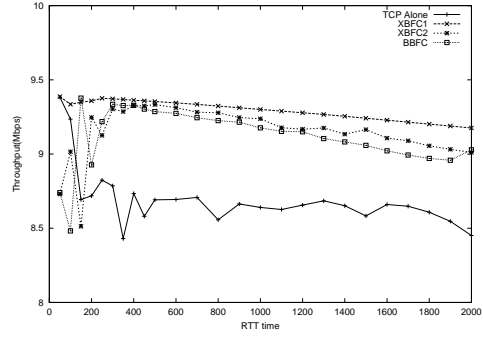


Figure 7: Total Throughput vs RTT

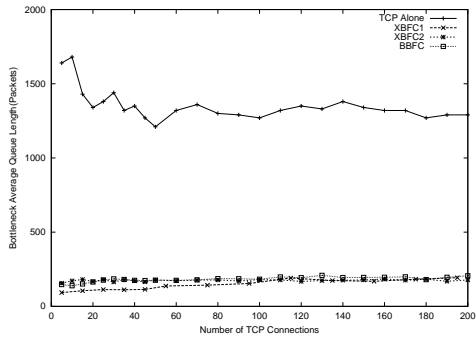


Figure 6: Average Queue Length vs Number of Connections

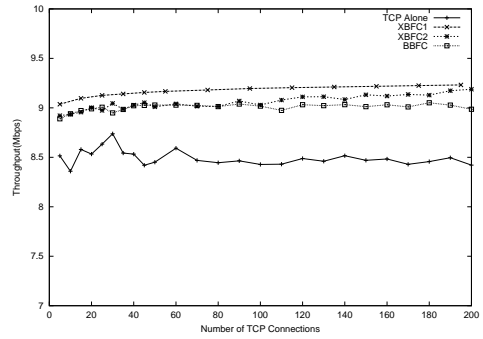


Figure 8: Total Throughput vs Number of Connections

On Figure 6), the  $x$ -axis represents the total number of TCP connections. Each point corresponds to one experiment. In each experiment,  $x$  long lived TCP connections with different round trip time share the bottleneck ( $i^{th}$  TCP connection has a one way propagation time of  $(50+i \times 2)ms$ ). At the end of the experiments, we report the average queue length experienced by all TCP connections.

First, we note again that **XBFC1**, **XBFC2**, and **BBFC** are almost indistinguishable.

Second, **TCP Alone** incurs a much higher average queue length than **XBFC1**, **XBFC2**, or **BBFC**.

Average queue length slightly decreases as the number of TCP connections increases. A higher contention on the wireless link seems to make TCP back off more often, leading to a slightly smaller average queue size.

### 6.3 Total Throughput

In this section, we report on Figures 7 and 8. the total throughput (in Mbps) obtained by all TCP connections during the same experiments performed in Section 6.2. The  $y$ -axis represents the total throughput of all TCP connections. On Figure 7, the  $x$ -axis represents the round trip

time in milliseconds. First, we note that **XBFC1** exhibits a slight advantage over **XBFC2** and **BBFC** for a round trip time higher than 400ms. For a round trip time between 50 and 400ms, **XBFC2** and **BBFC** exhibit a high variability in their performance. We are still investigating the reason why the performance for **XBFC2** and **BBFC** is so low and wildly changing in the range 50-400ms. Second, **TCP Alone**'s throughput is clearly lower than our schemes when the round trip time is higher than 400ms.

Figure 8 shows that **XBFC** slightly outperforms **XBFC2** and **BBFC**. Moreover, the total throughput for our schemes exceeds by 10% the throughput obtained with **TCP Alone**. The total throughput does not vary much with the number of TCP connections.

### 6.4 Packet Drop Probability

In this section, we present the packet drop probability for the same set of experiments as above.

Figure 9 presents our results for the packet drop probability *versus* the round trip time (in ms). First, note **XBFC1**, **XBFC2**, and **BBFC** almost never experience any packet drop while **TCP Alone** experiences drops, especially for low round trip times. With low round trip time, TCP con-

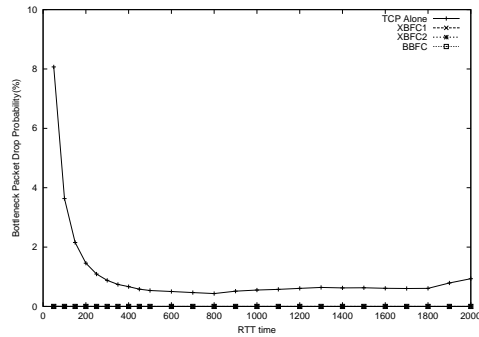


Figure 9: Packet Drop Probability vs RTT

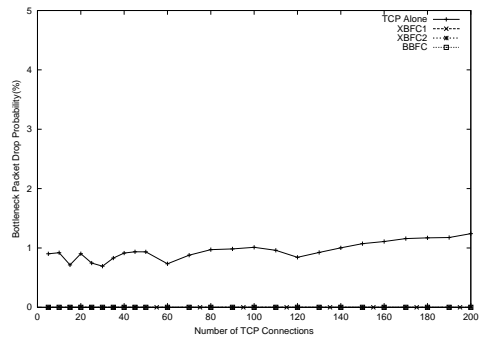


Figure 10: Packet Drop Probability vs Number of connections

nections reach high congestion window sizes. Moreover, since the propagation time is low, there are not many packets “on the wires”. Therefore, most packets have a high probability to get into the queue at the same time. This leads to high drop probability.

Figure 10 presents our results for the packet drop probability *versus* the number of TCP connections. The packet drop probability does not change much with the number of TCP connections.

In general, for all of our experiments, the three schemes **XBFC1**, **XBFC2**, and **BBFC** almost never experience any packet drops while TCP experiences as many as 10% of the packets.

## 7 Conclusion and Future Work

Our experiments show that **BBFC** can improve the throughput and the packet drop probability. **BBFC** yields also a much lower average queue size. Such observation may minimize the requirement space on basestations. But, the most remarkable property of **BBFC** is that it enforces fairness. Moreover, **BBFC** may be implemented without requiring any change to current TCP implementations. **BBFC** is very

promising because the extra information that TCP senders could provide does not bring any dramatic improvement: in all our experiments, **XBFC1** or **XBFC2** do not clearly outperform **BBFC**.

As future work, we want to study more closely **BBFC**. First, we want to set analytically the parameters  $\alpha$ ,  $\beta$ , and  $\gamma$  that yields a stable control algorithm. We also need to assess **BBFC** in a dynamic environment with a mix of short-lived and long-lived TCP connections.

One key problem is related to the variable available bandwidth on a wireless link: the link layer throughput is a function of the current contention on the wireless link and the channel condition. High contention on the medium and high bit error rate reduce the available bandwidth. We are currently considering this problem.

## References

- [1] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, “A comparison of mechanisms for improving TCP performance over wireless links,” in *ACM SIGCOMM’96*, pp. 152–159, Aug. 1996.
- [2] S. Biaz and D. Yawen, “Basestation flow control,” Tech. Rep. CSSE02-07, Computer Science and Software Engineering Department, Auburn University, Dec. 2002.
- [3] D.-M. Chiu and R. Jain, “Analysis of the increase and decrease algorithms for congestion avoidance in computer networks,” *Journal of Computer Networks and ISDN*, vol. 17, pp. 1–14, June 1989.
- [4] C. Fraleigh, S. Moon, B. L. Christophe Diot, and F. Tobagi, “Packet-level traffic measurements from a tier-1 IP backbone,” Tech. Rep. TR01-ATL-110101, SPRINT, Nov. 2001.
- [5] P. S. Henry, “Is Wi-Fi in your future?,” June 2002. Keynote Speech, IEEE Infocom 2002.
- [6] J.-H. Hu and K. L. Yeung, “FDA: A novel base station flow control scheme for TCP over heterogeneous networks,” in *IEEE Infocom’01, Anchorage, AL, USA*, Apr. 2001.
- [7] D. Katabi, M. Handley, and C. Rohrs, “Internet congestion control for future high bandwidth-delay product environments,” in *Proceedings of SIGCOMM’02*, ACM, Aug. 2002.
- [8] H. S. Martin, A. McGregor, and J. G. Cleary, “Analysis of internet delay times,” in *PAM 2000: Passive and Active Measurements Workshop, Auckland, New Zealand*, Apr. 2000.