FDA: A Novel Base Station Flow Control Scheme for TCP over Heterogeneous Networks

Jian-Hao Hu and Kwan L. Yeung
Department of Electrical and Electronic Engineering
The University of Hong Kong
Hong Kong, PRC.
Fax: (852) 2559-8738 E-mail: {jhhu, kyeung}@eee.hku.hk

***Abstract*: In this paper, a novel proactive base station flow control algorithm, called Forced Duplicate Acknowledgement (FDA), is proposed for extending TCP over wireless networks. FDA is implemented at the base station (BS) of a wireless network. It works in conjunction with existing TCP enhancement schemes such as Snoop or New Snoop. In FDA, if the average occupied buffer size at a BS exceeds a pre-defined threshold, an incipient congestion is detected. Then three forced duplicate ACKs, functioning as a congestion notification, will be generated by the BS and forwarded to a set of *selected* TCP senders. Upon receiving the forced duplicate ACKs, a sender reduces its transfer rate as a result of performing the fast retransmit procedure. To prevent multiple packet dropping due to buffer overflow at the BS, a quality guaranteed cache release policy is designed. The idea is to make room for the on-the-fly packets by releasing some cached but not-yet-acknowledged packets at the BS in advance. The performance of the proposed FDA is evaluated by simulations. We found that under various traffic and system configurations, FDA can not only fairly allocate the available wireless bandwidth among all TCP connections, but also achieves the highest throughput as compared to other schemes we have investigated.**

1. Introduction

Wireless data applications such as e-mail, web browsing, mobile computing, etc are gaining increased attention due to rapid advances in the areas of wireless communications and the Internet. Transmission Control Protocol (TCP) is an end-to-end reliable transport protocol in the Internet Protocol (IP) suite, which is widely used in popular applications like telnet, ftp, http, etc. TCP has been designed, improved and tuned to work efficiently in the wired network where the bit-error-rate is very low. Whenever a packet is lost, it is reasonable to assume that congestion has occurred on the connection path. Hence end-to-end congestion recovery algorithms, such as *fast retransmit* and *fast recovery* [11,13], are triggered to recover the loss efficiently.

In a heterogeneous network with both wired and wireless links, the assumption that packet loss signifies congestion is flawed because the wireless link has a much higher bit-error-rate, and a TCP connection might be temporally broken as a result of handoff. As a result, congestion recovery algorithms will be incorrectly triggered. This results in low link utilization and poor TCP performance in heterogeneous networks.

Since many network applications are built on top of TCP, it is necessary to enhance TCP performance in heterogeneous networks without (or with minimal) modifications to end-to-end TCP protocols. Various TCP enhancement schemes have been proposed for shielding the packet corruption on wireless links from the sender, including I-TCP [3], Link-level Retransmission [4], EBSN scheme [5], TCP SACK, SMART [6], and recently Snoop [1] and New Snoop [2]. As compared with other schemes, Snoop and New Snoop have the advantages of (i) keeping the end-to-end TCP semantics, (ii) compatible with the existing TCP applications, (iii) effectively shielding the TCP sender from wireless losses, and (iv) easy to implement. When a single TCP connection is supported by a base station (BS) implemented with Snoop or New Snoop, studies [1,2] showed that both of them give very good TCP performance. But we can show that their performance degrades significantly when multiple TCP connections are supported.

This is due to lack of a good flow control scheme at the BS. Because of the high bit-error-rate, narrow bandwidth, and high latency characteristics of the wireless link, a BS can easily become the bottleneck in an end-to-end TCP connection path. To make the situation worse, when Snoop or New Snoop is used, the buffer at the BS needs to store not only the packets waiting for transmission, but also the cached copies of the transmitted packets for local loss recovery.

Aim at solving the BS congestion problem, we propose a new BS flow control algorithm, called Forced Duplicate Acknowledgement (FDA), in this paper. Although FDA is designed to work in conjunction with Snoop or New Snoop, it can be generalized to any router that performs local packet loss recovery. Like Snoop and New Snoop, implementing FDA does not require TCP code modifications inside the fixed network, at the sender, or at the receiver. It should be noted that throughout this paper, we have assumed that all senders are located in the wired/fixed networks and all receivers are mobile hosts. Wireless Internet access is carried out via the BS.

In FDA, each BS monitors its average occupied buffer size. If a pre-defined threshold is exceeded, an incipient congestion is detected. Then three *forced* duplicate ACKs, functioning as a congestion notification, will be generated by the BS and forwarded to a set of selected TCP senders for flow control. Since the forced duplicate ACKs are generated by BS, it has a much quicker response time than using existing flow control algorithms like RED [8]. When congestion occurs at a BS, multiple packet dropping due to buffer overflow will freqently cause sender timeout, which in turn significantly reduces the TCP throughput. To prevent multiple packet dropping, a quality guaranteed cache release policy is designed. The idea is to make room for the on-the-fly packets by releasing some cached but not-yet-acknowledged packets at the BS in advance.

This paper is organized as follows. In the next section, the major difficulties facing TCP Tahoe and Reno, existing TCP flow control algorithms, and Snoop and New Snoop protocols, are summarized. Focusing on overcoming those difficulties, FDA scheme is proposed in Section 3. Its performance is evaluated in Section 4 using computer simulations. Finally we conclude the paper in Section 5. Because of the limited space, please refer to [11,13] for details about TCP Tahoe and Reno, and their congestion avoidance algorithms, fast retransmit and fast recovery. Please refer to [1,2] for details about protocols Snoop and New Snoop. Such knowledge is helpful in understanding the proposed FDA algorithm.

2. Major Difficulties

2.1 Problems with TCP Reno and Tahoe

The TCP incorporates a window-based congestion control algorithm to reduce congestion at end systems as well as at network routers. A TCP sender has a window's worth of outstanding/unacknowledged data, and it constantly probes network for more bandwidth and reacts to congestion via a dynamic window adjustment algorithm.

TCP Tahoe and TCP Reno [11,13] are the two most widely used TCP versions in the Internet. Due to the narrow bandwidth and high latency at the wireless link, network congestion may frequently occur at the base station (BS) and this would result in multiple packets being dropped at the BS during a congestion episode. If three or more consecutive packets are lost, sender timeout must occur in order to recover the loss. Even without causing sender timeout, the high packet loss rate at the wireless link would trigger frequent fast retransmit and fast recovery at the sender. This causes *SSTHRESH*, the estimate of the steady state network capacity, shut down to a very small value. The corresponding TCP sender is then forced to transmit at a very low rate, and the wireless bandwidth is wasted. It can be shown that if transmission window size $CWND < 4$, the TCP sender cannot recover from the congestion state but to timeout.

From the above, we can see that to achieve high throughput at wireless link, a BS should (i) shield the senders from packet losses at the wireless link, and (ii) avoid BS congestion and thus multiple packet dropping due to BS buffer overflow. Snoop [1] and New Snoop [2] can deal with the first problem. The focus of this paper is on the second problem.

2.2 Challenges for TCP flow controls

TCP flow control algorithms face two problems: *slow reaction to congestion* and *biased against connections with long RTTs* (round-trip-times). These problems become more serious in heterogeneous networks due to the narrow bandwidth and high latency of the wireless links. When congestion occurs, packets from almost all connections will be dropped due to buffer overflow. This subsequently triggers all senders to reduce their transmission rates (i.e. transmission window size) at almost the same time, thereby causing an abrupt drop in throughput. This problem is known as *global synchronization* [14]. In response to the congestion, each sender-receiver pair will gradually open up its transmission window. Since the RTT between each sender-receiver pair is different, connections with long RTTs will open up their transmission window sizes at a much slower speed than those with short RTTs. Severe unfairness is likely as the short RTT connections will grab the available bandwidth well before the long RTT connection can.

Some flow control algorithms have been proposed to address the above two problems. Among them, random early detection (RED) [8], back-pressure [9] and constant-rate (CR) [7] are widely studied. In CR [7], the sender increases its window size according to its RTT during congestion avoidance phase. This can correct the bias against the connections with long RTTs. But it is difficult to implement since (i) it is hard to decide the sender parameters for different network topologies, and (ii) it needs to modify the TCP sender congestion control algorithm. The back-pressure scheme [9] focuses on flow control in the backbone network. It assumes that ATM available bit rate connections will be used for flow control of individual TCP sessions. For this, we believe it is not likely to happen. A similar study can be found in [10].

Unlike CR and back-pressure algorithms, RED [8] takes a proactive approach. It slows down TCP senders with early congestion detection. When the average queue length at a bottleneck router exceeds a threshold, RED starts to randomly drop the incoming packets. From [8], RED achieves good performance in reaction time and fairness if the number of connections through the bottleneck router is not too large. Global synchronization can also be avoided.

For heterogeneous networks with both wired and wireless links, direct applying RED and its variants to BS implemented with Snoop or New Snoop, is unfortunately not feasible. This is because the buffer at a BS is shared by both packets waiting for transmission, and packets waiting for acknowledgements (ACKs). A cached packet is released from the buffer only when an ACK that confirms the correct reception by the mobile host arrives (refer to [1,2] for details). The congestion notification (by packet dropping) in RED could lead to more unacknowledged packets to be cached at the BS. This in turn causes more packets to be dropped there. This forms a positve feedback loop and the congestion will collapse. We call this phenomenon *congestion expansion*. Congestion expansion can only be recovered by sender timeout and is therefore very inefficient and undesirable. The effect of congestion expansion can be observed from the simulation results in Section 4.

2.3 Challenges for Snoop and New Snoop

By adopting a TCP-aware link retransmission scheme, both Snoop and New Snoop enhance the TCP performance without sacrificing the end-to-end TCP semantics, re-linking existing applications at the BS, and modifying host TCP codes inside the fixed network, at the sender or receiver.

Earlier studies [1,2,12] of Snoop and New Snoop focus on supporting a single TCP connection. The congestion control policy in New Snoop[1] [2] works well for one connection but becomes inefficient for multiple connections. Suppression of duplicate ACKs for packets cached at BS buffer can temporarily stop the sender's transmission, but it cannot adjust

[1] Original Snoop [1] protocol does not have a flow control algorithm.

the senders' transmission window size according to the wireless link traffic loading. As a result, many packets from the long RTT connections would be dropped in the slow start phase, and global synchronization would cause their throughputs to drop significantly.

In TCP, the traffic injected by the sender into the network is bursty as several outstanding packets at the sender can be acknowledged by a single new ACK. For a heterogeneous network, TCP traffic tends to become more bursty because of the long delay and poor quality at the wireless link, as well as the local loss recovery mechanisms adopted by Snoop and New Snoop. As a result, when a burst of packets arrive at the BS during a congestion episode, multiple packet dropping due to buffer overflow will frequently occur. This creates another major challenge for supporting multiple TCP connections.

3. Forced Duplicate ACK (FDA) Scheme

Aim at meeting the challenges summarized in the previous section, a novel base station flow control scheme, called Forced Duplicate Acknowledgement (FDA), is proposed in this section. The whole idea of FDA is based on manipulating the received acknowledgement (ACK) stream from the mobile hosts. The flowchart in Fig. 2 summarizes the scheme. It should be noted that parameters (P_a, P_c, P_r, P_s and *Threshold*) in the flowchart are obtained in the system design phase, as detailed in this section. Once they are obtained, they can be stored in the data table at the BS during operation. Since FDA is designed to work in conjunction with either Snoop or New Snoop, for ease of presentation, from now on we will use snoop to mean either Snoop or New Snoop. When we refer to a particular protocol, we will use Snoop or New Snoop.

3.1 FDA Scheme Overview

Let the buffer at a BS be shared by all TCP connections as shown in Fig. 1. Two types of packets can be found in the buffer, packets waiting for transmission and packets waiting for acknowledgements (also called cached packets). The operation of FDA consists of three important functions:

- *Incipient Congestion Detection*,
- *Congestion Notification Generation*, and
- *Quality Guaranteed Cache Release*.

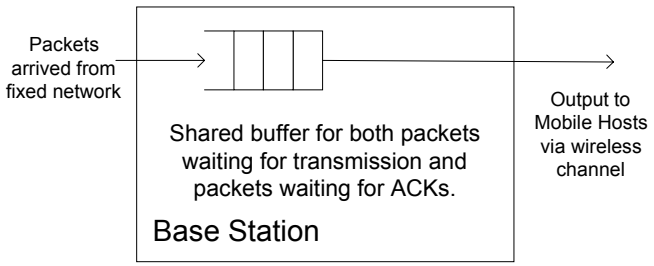

Figure 1. Shared buffer at a BS.

A. Incipient Congestion Detection

Each BS monitors its average queue length, i.e. average occupied buffer size. When the average queue length (*avg*) exceeds a pre-defined congestion threshold (*Threshold*), an incipient congestion is detected, meaning that if all connections remain transmitting at their current rates, congestion at the BS would soon occur. Similarly, when the *instantaneous* queue length exceeds *Threshold*, we say a *transient incipient congestion* is detected.

When a packet is corrupted at the wireless link, snoop will be activated for local packet loss recovery. The queue at the BS builds up quickly as no *new* ACK (i.e. not a duplicate ACK) will be received for acknowledging (and thus releasing) the cached packets in the buffer. A transient incipient congestion occurs. If the transient incipient congestion persists for some time, it becomes an incipient congestion if $avg \geq Threshold$. In this case, the incipient congestion is said to be caused by local loss recovery mechanism of snoop, *not* because of total incoming traffic at the BS is too high. Therefore, a good congestion detection algorithm should be able to identify the cause of the incipient congestion. If it is due to the local loss recovery, no congestion notification should be generated. Otherwise, congestion notification (as described in the next function) should be generated as quickly as possible.

In Section 3.2, a method for calculating average queue length *avg* is proposed. Together with the congestion threshold *Threshold* derived in Section 3.3, an accurate incipient congestion detection function can be obtained such that the false triggering of flow control actions can be minimized.

B. Congestion Notification Generation

When an incipient congestion is detected, the BS activates the function for *congestion notification generation*. The purpose is to select a subset of current active TCP connections via the BS for flow control. Once identified, congestion notification messages, in the form of three forced duplicate ACKs, will be generated by the BS and forwarded to their corresponding senders for slowing down their transfer rates.

In RED, congestion notification is achieved implicitly by randomly dropping some incoming packets at the BS/router with a probability derived from the current traffic loading. When a receiver detects the loss after some time, duplicate ACKs for the dropped packets will be generated and forwarded to the sender. When three consecutive duplicate ACKs arrive at the sender, the sender slows down its transfer rate as a result of performing fast retransmit. Because of the congestion expansion problem discussed in Section 2.2, RED cannot be applied when snoop is used.

In FDA, congestion notification, again in the form of three duplicate ACKs, is generated by the BS instead of the receiver. (Therefore we call them *forced* duplicate ACKs.) The congestion response time is faster than using RED and there will be no packet dropping at the BS. This helps to prevent congestion expansion. To determine which connection should be selected for flow control, we follow a probabilistic approach based on the received new ACK stream from the receiver. When a new ACK arrives at the BS, three duplicate copies of it will be generated with a pre-determined probability P_b, called notification generation probability. Its value is derived in Section 3.4. This probabilistic approach is fair in the sense that a connection occupied a larger portion of the wireless bandwidth will be selected with a higher probability.

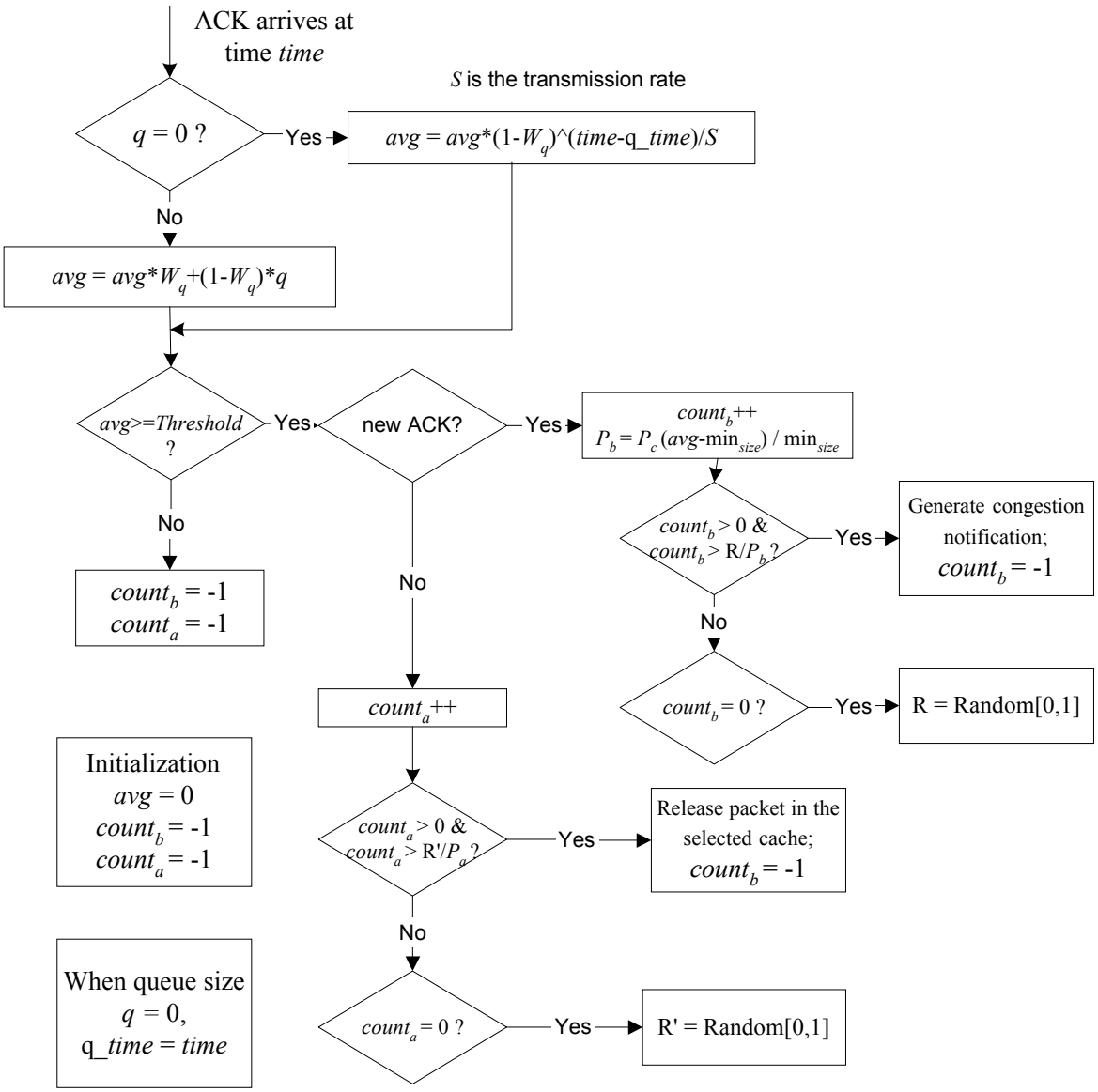

Fig. 2 Flowchart of FDA algorithm.

Generating congestion notification based on new ACK stream can help to reduce the traffic burstiness. Recall that when a new ACK arrives at the sender after a sequence of duplicate ACKs, due to the sudden opening of the transmission window size, a burst of packets will be injected into the network. If this new ACK is followed by three forced duplicate ACKs, then the sender will immediately stop its packet injection and perform fast retransmit. As a result, the length of the packet burst is reduced. This in turn helps to reduce the chance of multiple packet dropping later on when these packets arrive at the BS. (To further reduce the chance of multiple packet dropping, quality guaranteed cache release function is designed in the next section.) Another reason of not generating congestion notification based on duplicate ACKs is because using snoop [1,2], duplicate ACKs will be held by the BS to avoid erroneous retransmission at the sender.

When the three forced duplicate ACKs arrive at the sender, the sender enters the fast retransmit procedure and the requested packet is retransmitted. If the next arrived ACK (i.e. immediately after the three duplicate ACKs) is a new ACK, the sender enters a new steady state. In the new steady state, the sender will be idle for *at most* half of its round-trip-time before it can transmit any new packet at the half rate of the original state. If the next arrived ACK at the sender is again a duplicate one, this duplicate ACK *must* be generated by the receiver. Then the sender remains in the fast retransmit mode until a new ACK is received. At the receiver side, when the retransmitted packet arrives, it will be dropped if the receiver has already correctly received it.

C. Quality Guaranteed Cache Release

Quality guaranteed cache release function is designed to minimize the chance of multiple packet dropping at a BS during a congestion episode. Consider a worst case scenario that the buffer at a BS is fully occupied. In this case, the BS can receive only duplicate ACKs from the receiver because otherwise each new ACK would trigger some cached packets to be released and thus the buffer will not be full. If the packet indicated by the duplicate ACKs cannot be found in the buffer, it must be lost in the fixed network. The received duplicate

ACKs are then forwarded to the sender for triggering sender retransmission.

According to the fast retransmit and fast recovery algorithms in TCP Tahoe and Reno, the sender will transmit new packets during a congestion episode when its *CWND* size is opened sufficiently large. Also due to the bursty nature of the traffic injected by the sender, very often that when the BS buffer is full, there are still some on-the-fly packets in the fixed network, moving toward the BS. When they arrive, they will be dropped due to buffer overflow. This results in the very undesirable multiple packet dropping. To make the situation worse, some of the dropped packets could be retransmitted repair packets from the sender.

On the other hand, some cached but not-yet-acknowledged packets in the BS buffer are *useless* in the sense that they have already been correctly received by the receiver as *out-of-order packets*. If we can release them in advance (i.e. before their corresponding ACKs arrive), then we can make room for those more critical on-the-fly packets. The problem is how to identify those *useless* packets. According to the ACK generation mechanism, a receiver generates an ACK (either new or duplicate)[2] for each correctly received packet. Therefore whenever an ACK arrives at the BS, we know for sure that one packet has been correctly received by the corresponding receiver. Since we do not know which particular packet has been correctly received, a cache release probability P_a is derived in Section 3.5 for releasing a cached packet on a probabilistic basis.

An *erroneous cache release* occurs if an unacknowledged packet is lost at the wireless link and its cached copy has been (erroneously) released in advance. The retransmission of this packet can only be performed by the sender. This will slightly degrade the end-to-end TCP performance but as shown by our simulations in Section 4, such impact is not significant. In order to reduce the number of erroneous cache release, quality guaranteed cache release policy is triggered only when the average queue length exceeds the congestion threshold.

Combining the above three functions, *incipient congestion detection*, *congestion notification generation*, and *quality guaranteed cache release*, FDA can be summarized by the following pseudo codes.

For each ACK arrival
  / Incipient Congestion Detection Function */*
  Calculate the average queue length, avg (Section 3.2)
  If avg ≥ Threshold
    If ACK is a new ACK
    / Congestion Notification Generation Function */*
      Calculate probability P_b (Section 3.4)
      With probability P_b
        Generate three forced duplicate ACKs and forward them to the corresponding TCP sender
    Else
    / Quality Guaranteed Cache Release Function */*
      Calculate probability P_a (Section 3.5)
      With probability P_a
        Select one cached packet from the corresponding connection and release it from the buffer.

3.2 Average Queue Length Calculation

Like RED algorithm, FDA uses a low pass filter to find the average BS queue length *avg*, i.e.

$$avg = (1 - W_q)avg + W_q q \quad (1)$$

where W_q is the time constant of the low pass filter, and q is the instantaneous queue length at the time of computing. The value of W_q determines the responsiveness of the average queue length to the instantaneous queue length changes. If W_q is close to 0, the average queue length responds very slowly to the queue length changes. In this case, the congestion may have occurred before any flow control action can be taken. If W_q is close to 1, the system becomes too responsive and the averaging procedure in (1) will not be able to filter out the transient incipient congestion at the BS. A transient incipient congestion is usually caused by recovering *isolated* packet loss at the wireless link. A wireless packet loss is isolated if

- at the time the loss is detected by the receiver, all previous packet losses on the same connection, if any, have already been recovered, and
- from the moment the loss is detected to the moment that a retransmitted repair is received, there is no other packet losses.

It can be seen that for a typical range of packet loss rates from 10^{-1} to 10^{-4}, most packet losses at the wireless link are isolated. (This is, however, generally not true for packet losses inside fixed network or at the BS, where those losses are caused mainly by congestion/buffer overflow.)

Next we need to find a suitable value for W_q. Let the maximum number of packets that can be transmitted in the wireless round trip delay time between the BS and its furthest away (mobile) receiver at the *full* wireless link speed be $\min_{size}$. $\min_{size}$ is also the minimum number of packets that the BS should cache in order to recover an isolated packet loss. Assume the system is in a steady state with no packet loss/ congestion for a sufficiently long time. The aggregated transfer rate of all connections via the BS will be equal to the transfer rate of the wireless link, and the average queue length at the BS will be roughly equal to $\min_{size}$. At BS, each new ACK clocks out a cached packet from the queue; and each new packet increases the queue length by one. The average *net* queue length change is 0. For simplicity, we assume that both instantaneous queue length q and average queue length *avg* are of same value $\min_{size}$ at this steady state.

Without loss of generality, let an isolated packet loss occur on a particular connection. When duplicate ACK generated by the corresponding receiver arrives at the BS, *no* cached packet of that connection can be released (unless the quality guaranteed cache release function is used). The queue length at the BS will be increased from $\min_{size}$ to $N + \min_{size}$ if N new packets of that connection arrive at the BS subsequently. When the N-th packet arrives, the average queue length is given by:

[2] A New ACK is generated if the received packet's sequence number is in sequence with respect to the last received one. Otherwise, a duplicate ACK is generated.

$$avg_N = (1-W_q)^N \min_{size} + \sum_{i=1}^{N} (i + \min_{size}) W_q (1-W_q)^{N-i}$$

$$= \min_{size}(1-W_q) + N + 1 + \frac{(1-W_q)^{N+1} - 1}{W_q} \qquad (2)$$

When the repair of the lost packet arrives at the receiver, a new ACK is generated. When it arrives at the BS, all cached packets of that connection with sequence numbers smaller than that of the new ACK will be flushed immediately, causing a sudden drop in BS queue length.

Given that the congestion threshold is *Threshold* and there are $\min_{size}$ outstanding packets in the BS buffer. If we wish to allow a burst of L packets arriving at the BS without triggering the (false) congestion notification generation function, the time constant W_q in (2) should be chosen to satisfy $avg_L < Threshold$, or

$$\min_{size}(1-W_q) + L + 1 + \frac{(1-W_q)^{L+1} - 1}{W_q} < Threshold \qquad (3)$$

Given the values of L and *Threshold*, we can get the upper bound on W_q from (3). In FDA, W_q is set to its upper bound in order to get a short response time. In this paper, we set the value of L to be equal to the initial value of *SSTHRESH*.

3.3 Congestion Threshold and BS Buffer Size Selection

It is difficult to decide the optimal value for *Threshold* because it depends on many factors, such as the bursty nature of the traffic, the wireless packet loss rate, and the number of connections sharing the buffer. For a typical range of packet loss rates from 10^{-1} to 10^{-4}, most packet losses at the wireless link are isolated and can usually be recovered by a single local retransmission. With that in mind, we suggest to set

$$Threshold \geq 2 \times \min_{size}. \qquad (4)$$

If $Threshold = 2 \times \min_{size}$, recovering an isolated packet loss will not cause the queue length (both instantaneous and average) to exceed *Threshold*. This helps to make sure that the transient incipient congestion caused by isolated packet loss will not lead to a false alarm of incipient congestion. For simulations in Section 4, we adopt *Threshold* to be just larger than $2 \times \min_{size}$.

Next we provide some guidelines in designing the appropriate buffer size at a BS. For a good flow control scheme, the aggregated transfer rate of all connections via the BS should be kept around the transfer rate of the wireless link, and the average queue length should be equal to $\min_{size}$, when there is no packet loss at wireless link. Assume a sender always has packets to transmit and its transfer rate equals to the rate of the wireless link. Let the maximum number of local retransmissions required for recovering a corrupted packet on the wireless link be n. Then the BS buffer size *Total_buffer_size* should store at least $(n + 1)\ \min_{size}$ packets.

Since quality guaranteed cache release function is adopted, we should take its effect into consideration. When quality guaranteed cache release function is used, let $(1-P_r)$ be the *given* probability requirement that a packet loss at the wireless link can be found at the BS. Let P_s be the packet loss probability at the wireless link. We suggest the BS buffer size *Total_buffer_size* to satisfy the following condition:

$$Total\_cache\_size \geq \left(\frac{\log P_r}{\log P_s} + 1 \right) \times \min_{size} \quad \text{and}$$

$$Total\_cache\_size \geq Threshold \geq 2 \min_{size} \qquad (5)$$

The above inequalities only provide a guideline for BS buffer size design. The idea is to relate the number of local retransmissions required to probabilities P_r and P_s. This can help to prevent the possible buffer size over-engineering. For our simulations in Section 4, where $P_r = 10^{-3}$ and $P_s = 10^{-1}$ to 10^{-4}, we set *Total_buffer_size* = 128 packets and this value (just) satisfies the above inequality. We can demonstrate that this setting gives a good performance.

3.4 Probability for Generating Forced Duplicate ACKs

When *avg* exceeds *Threshold*, incipient congestion is detected at the BS. The congestion notification generation function is then activated for selecting a set of TCP connections for flow control. As discussed earlier, for each received new ACK, three duplicate ACKs for that connection will be generated with a given probability P_b, called notification generation probability.

When *avg* = *Threshold*, let P_c be the minimum probability of generating a congestion notification on a received new ACK. (In Section 4, $P_c = 0.016$ is used.) It is natural to have a higher notification generation probability if *avg* increases beyond *Threshold*. Therefore we design P_b for each new ACK, as a function of average queue length, where

$$P_b = P_c \cdot (avg - \min_{size}) / \min_{size}. \qquad (6)$$

In order to avoid notification message clumping, BS should generate notifications at a regular interval (i.e. evenly spaced among received new ACKs). Let the notification generation interval be every X new ACKs. It is desirable to make X a uniform random variable between 1 and $1/P_b$, or $[1, 1/P_b]$. To achieve this, the notification generation probability for each new ACK becomes $P_b/(1 - count_b \cdot P_b)$, where $count_b$ is the number of new ACKs that have arrived since the last notification generation. Hence we have

$$P(X = n) = \frac{P_b}{1-(n-1)P_b} \prod_{i=0}^{n-2} \left(1 - \frac{P_b}{1 - i \cdot P_b}\right) = P_b, \text{ for } 1 \leq n \leq 1/P_b$$

$$P(X = n) = 0 \qquad \text{for } n > 1/P_b \qquad (7)$$

The expected notification interval is $E[X] = 1/(2P_b) + 1/2$.

In Fig. 2, a simpler method for generating congestion notification is adopted. Instead of generating a unique random number for each received new ACK and comparing it with $P_b/(1 - count_b \cdot P_b)$, we randomly choose one new ACK from each ACK interval $[1, 1/P_b]$. This can greatly simplify the implementation complexity while without sacrificing the performance. The same approach is adopted in designing RED [8] algorithm. As a side remark, maximum queue length threshold $\max_{th}$, as defined in RED, is not needed in FDA. For RED, when average queue length exceeds $\max_{th}$, all arrival packets will be dropped with probability 1.

3.5 Probability for Quality Guaranteed Cache Release

When average queue length *avg* exceeds *Threshold*, the quality guaranteed cache release function is activated. Recall that $(1-P_r)$ is defined as the probability requirement for a repair to be found in BS cache, and P_s is the packet loss probability at wireless link.

If $P_s > P_r$, set $P_a = P_r/P_s$. That is for each *duplicate* ACK received, the BS releases a randomly selected cached packet from the connection identified by this ACK with probability $P_a/(1-count_a \cdot P_a)$, where $count_a$ is the number of duplicate ACKs that have arrived since the cache release function has been activated. Same as in generating forced congestion notification in Section 3.4, probability $P_a/(1-count_a \cdot P_a)$ is used to make sure that the selected duplicate ACK is evenly spaced in the received duplicate ACK stream.

If $P_s \le P_r$, set P_a=1. That is for each duplicate ACK received, release a randomly selected cached packet with probability 1. For implementation, a simpler method as shown in Fig. 2 is used to determine the cache release event.

It should be noticed that in selecting a cached packet for advance releasing, those packets that have been marked as *retransmission packets* using snoop [1,2] protocol are not supposed to be chosen.

4. Simulation Results

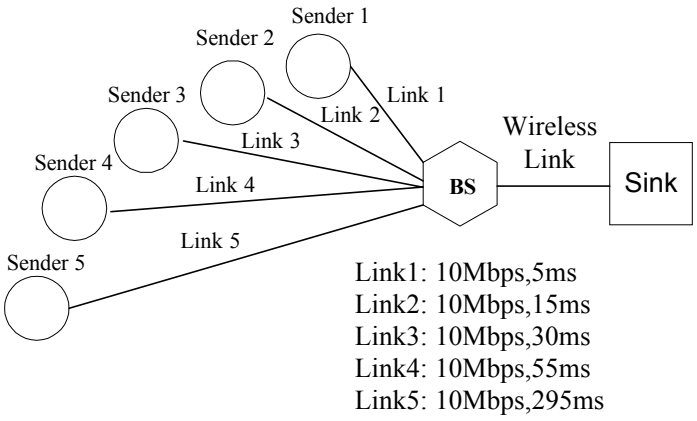

Fig. 3 Network model A with five TCP connections

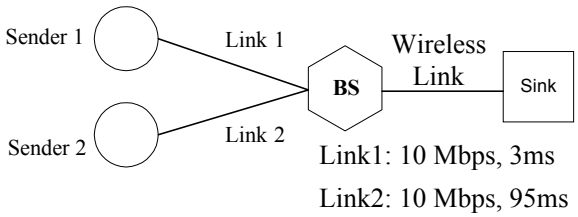

Fig. 4 Network model B with two TCP connections.

In this section, we evaluate the performance of FDA together with Snoop and New Snoop protocols by simulations. Two network models, as shown in Figs. 3 & 4, are simulated. We assume the senders always have packets to send during the whole simulation time. The sink receives all TCP packets and generates the corresponding ACKs for each connection. The TCP version used is TCP Reno. The performance metrics adopted are the throughput of the wireless link, fairness among all connections and BS buffer utilization. Their definitions are:

$$Throughput = \frac{total\_packet \cdot packet\_size}{wirelesslink\_rate \cdot simulation\_time} \quad (8)$$

$$Fairness = \frac{(\sum_{i=1}^{n} b_i)^2}{n \cdot \left(\sum_{i=1}^{n} b_i^2\right)} \quad (9)$$

$$Utilization = \frac{average\_used\_buffer\_size}{Total\_buffer\_size} \quad (10)$$

where b_i is the fraction of the bandwidth taken up by connection i on the wireless link. The redundancies of packet/frame construction and physical layer coding are not considered in the throughput calculation. ACKs are assumed not experiencing any loss because of their small packet size. From (9), we can see that the value of *Fairness* ranges from $1/n$ to 1, with 1 corresponding to equal/best allocation among all connections.

Wireless links with two different link speeds are simulated: 2 Mbps high bandwidth link (e.g. wireless LAN and the 3rd generation mobile systems) and 76.8 kbps low bandwidth link (e.g. GPRS). The main simulation parameters are listed in Table 1. The minimum cache size min_{size} is calculated from the round trip time of the wireless link, which is found to be smaller than but close to 30 packets. For simplicity, we fix min_{size} = 30 packets for all simulations. Accordingly, the value of *Threshold* is chosen to be 2 min_{size}, or 60 packets. In our implementations of both Snoop and New Snoop protocols, a fixed BS buffer size of *Total_buffer_size* = 128 packets is shared by both packets waiting for transmission and packets waiting for ACKs.

Table 1 Simulation Parameters

Parameters	Symbols	Values
The prob. of FDA to generate congestion notification.	P_c	0.016
The requirement parameter for quality guaranteed cache release	P_r	0.001
Low pass filter time constant	W_q	0.022
The prob. of packet loss in wireless link	P_s	10^{-1}~10^{-4}
Congestion threshold for FDA	*Threshold*	60 pkts
The total buffer size at the BS	*Total_buffer_size*	128 pkts
The burst traffic size	*L*	64 pkts
Initial steady state bandwidth estimation	*SSTHRESH*	32Kbyte
The receiver window size per connection at sink	*RCVWND*	64Kbyte
The round trip time of wireless link	$RTT_{process}$	50ms, 1500ms
The transfer rate of wireless link	Wireless_rate	2Mbps, 76.8kbps
TCP packet size	Packet_size	512 Bytes
ACK size	ACK_size	40 Bytes

4.1 Using 2Mbps High Bandwidth Wireless Link

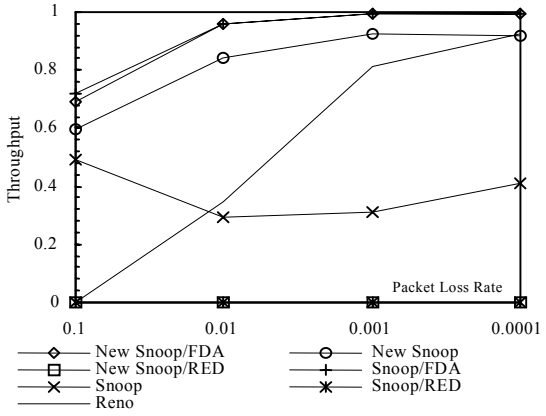

Fig. 5(a) Model A: Throughput.

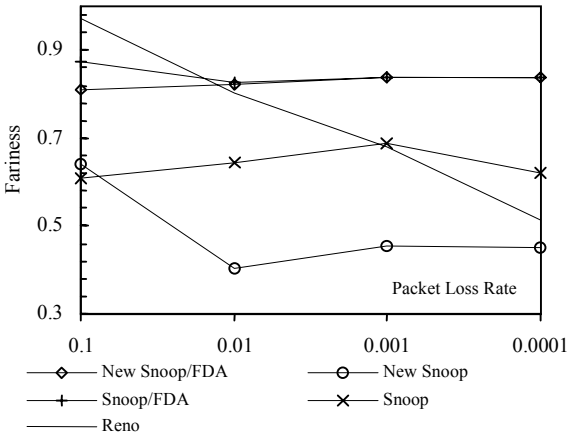

Fig. 5(b) Model A: Fairness.

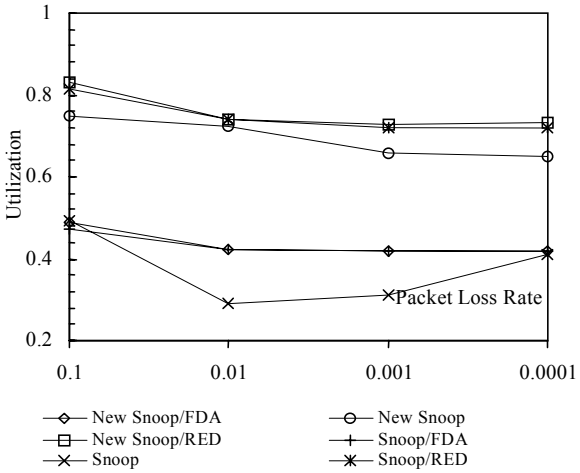

Fig. 5(c) Model A: Buffer Utilization

First we consider the case of using 2 Mbps wireless link. The systems shown in Figs. 3 & 4 are simulated for 1 hour each to get a stable performance. The RTT of wireless link is 50 ms according to current wireless LAN specifications [1]. For comparison, two types of TCP algorithms are simulated: (i) *without BS flow control*: (pure) Reno, Snoop and New Snoop, and (ii) *with BS flow control*: Snoop/RED (i.e. Snoop-with-RED), New Snoop/RED, Snoop/FDA and New Snoop/FDA. When RED is used, the BS functions like an ordinary router in the fixed network with a FIFO buffer size of 128 packets.

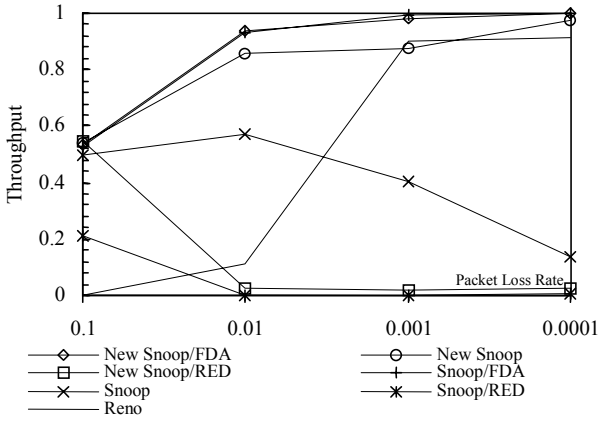

Fig. 6(a) Model A: Throughput.

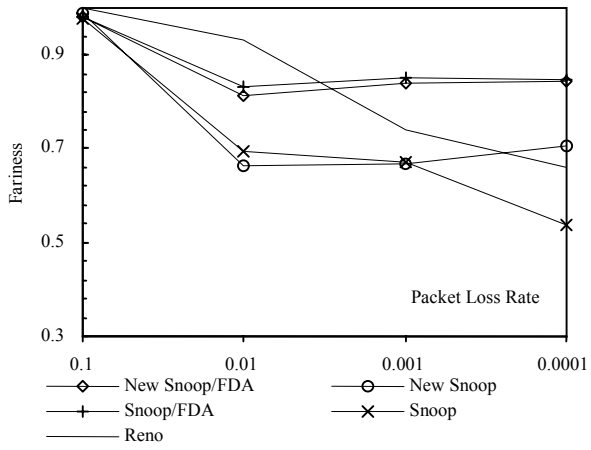

Fig. 6(b) Model B: Fairness

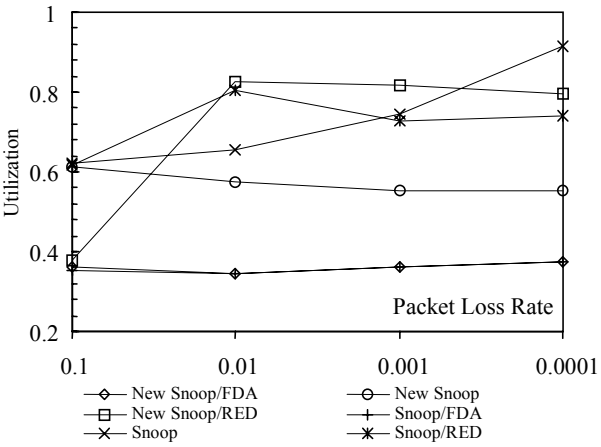

Fig. 6(c) Model B: Buffer Utilization

Figs. 5 & 6 show the performance results using network models A and B respectively. From throughput performance in Figs. 5(a) & 6(a), we can see that Snoop/FDA and New Snoop/FDA give almost the same highest throughputs. Because of the high packet loss probability at wireless link, the throughput of (pure) Reno is rather low. Due to congestion expansion, the throughputs obtained by Snoop/RED and New Snoop/RED are even lower that of Reno. Therefore this confirms our earlier conjecture that RED should not be used at the BS together with Snoop or New Snoop. Since New Snoop takes the congestion at BS into account [2], while Snoop

suffers from multiple packet dropping due to buffer overflow, New Snoop can achieve a higher throughput than that of Snoop. We also find that the throughput of FDA in model A is a little bit higher than that in model B because model A carries a larger number of TCP connections.

From Figs. 5(b) & 6(b), we can see that for both Snoop/FDA and New Snoop/FDA, their *Fairness* values are consistently higher than other algorithms, and are all above 0.8 for various packet loss rates. That means the wireless bandwidth is fairly shared among all connections. Due to the bias against connections with long RTTs, when Reno, Snoop or New Snoop protocol is used, the wireless bandwidth is almost completely grabbed by connections with small RTTs. For RED, since its throughput is so low that its fairness performance is meaningless and thus not shown in Figs. 5(b) & 6(b).

From Table 1, we find that the value of *Threshold* is about 46.7% (=60/128) of the total buffer size. From buffer utilization performance in Figs. 5(c) & 6(c), we can see when FDA is used, the average queue length is kept very close to *Threshold* and leaves about half of the BS buffer space for (i) isolated packet loss recovery, and (ii) storing possible bursty incoming traffic. We also find that the buffer utilization of FDA in model A is higher than that in model B. Due to congestion expansion at the BS and the frequent timeouts at the sender, using RED has a very low throughput (as shown in Figs. 5(a) & 6(a)) but a high buffer utilization.

To summarize, since flow control is performed according to traffic loading and packet loss rate, the fairness and buffer utilization performances of FDA are robust for various packet loss rates. Because Snoop and New Snoop protocols have adopted similar methods to recover packet losses at wireless link, they have a comparable performance when FDA is adopted. We also find that the performance of FDA is robust for the number of TCP connections carried.

4.2 Using 76.8 kbps Low Bandwidth Wireless Link

Next we study the performance of FDA in systems with a slow speed wireless link of 76.8 kbps. Due to the widely used narrow-band digital mobile systems, it is important for FDA to function well in supporting GPRS-like services. In this case, the simulation time is increased to 10 hours for getting a stable performance. The RTT of the wireless link is 1500 ms according to [12]. Like the previous case, various TCP enhancement schemes are simulated for comparison. The performance results are shown in Figs. 7 and 8 for network models A and B respectively.

From throughput performance in Figs. 7(a) and 8(a), we can see that both Snoop/FDA and New Snoop/FDA can achieve the highest throughputs. Unlike the case with 2Mbps wireless link, the throughput of New Snoop with/without RED is close to that with FDA. This is because the major component in end-to-end RTT is the slow wireless loop. As a result, most outstanding packets are waited at the BS. Since New Snoop [2] can stop the sender's transfer rate by suppressing the duplicate ACK for packets corrupted in the wireless link when local recovery occurs, New Snoop does not suffer from the serious multiple packet dropping problem, and thus no congestion expansion at the BS. This is not true for Snoop. Therefore, when Snoop with/without RED is used, congestion expansion occurs, which results in a very low throughput.

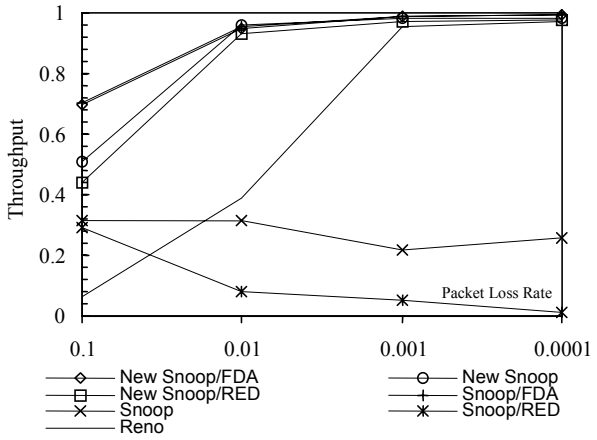

Fig. 7(a) Model A: Throughput.

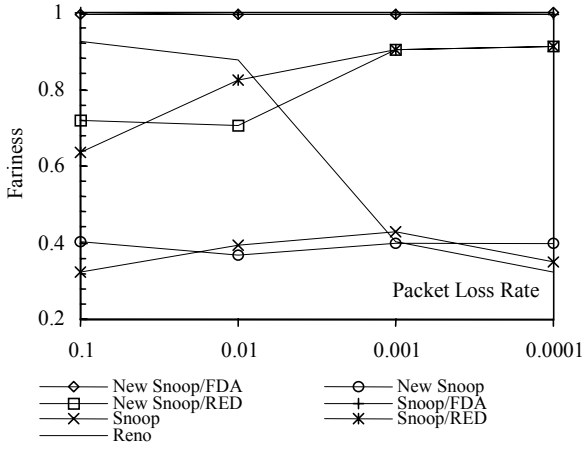

Fig. 7(b) Model A: Fairness.

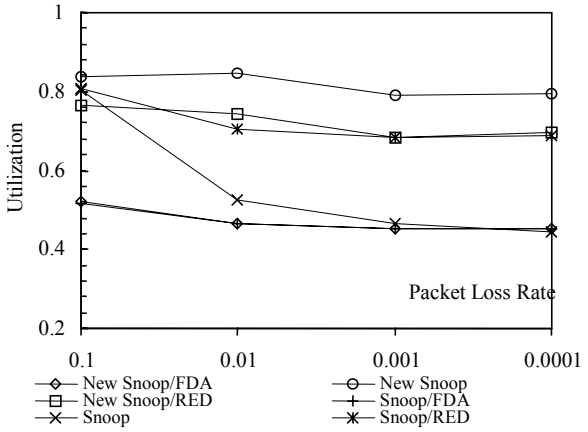

Fig. 7(c) Model A: Buffer Utilization

Also due to the long RTT at the wireless loop, the Fairness values of FDA shown in Figs. 7(b) & 8(b) are close to 1. On the other hand, it is reasonable to believe that for the three schemes without BS flow control, Reno, Snoop and New Snoop, similar excellent fairness performance could be achieved. However, this is not the case because the bandwidth of wireless link is much narrower than that of the wired links. When the simulation starts, all senders work in slow start

modes and the BS buffer overflows very quickly. Multiple packets from some farther away senders will be frequently dropped, and the retransmissions of those packets can only be triggered by sender timeout. As a result, the throughputs of those connections are always low during the whole simulation period. In other words, the global synchronization occurs. That explains why Reno's fairness is excellent for high packet loss rate, while poor for low packet loss rate. For Snoop and New Snoop protocols, although they shield the senders from packet loss at the wireless link, multiple packet dropping during slow start phase is serious. Therefore their fairness performance is also poor.

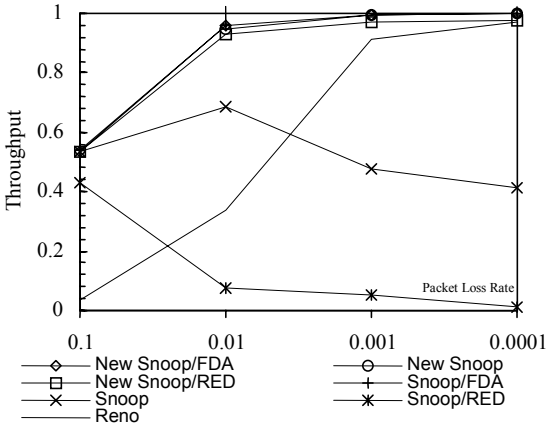

Fig. 8(a) Model B: Throughput

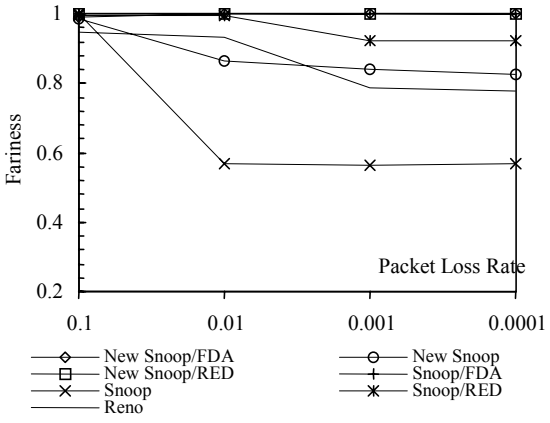

Fig. 8(b) Model B: Fairness

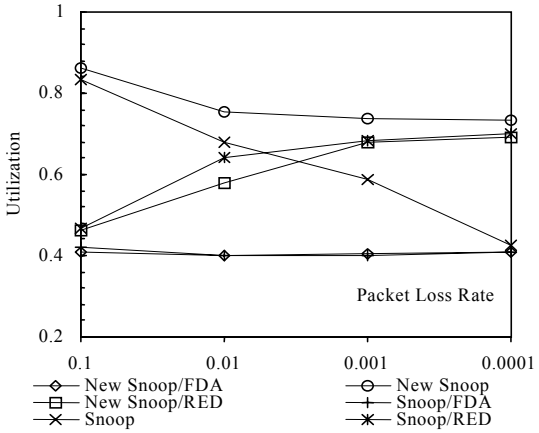

Fig. 8(c) Model B: Buffer Utilization

Finally from buffer utilization performance in Figs. 7(c) & 8(c), we can again find that FDA keeps the average queue length at the BS close to *Threshold* and leaves about half of the BS buffer for isolated packet loss recovery and burst traffic handling. Overall speaking, we find that FDA gives a consistent good performance regarding throughput, fairness and BS buffer utlization when a narrower wireless link is used.

5. Conclusions

In this paper, a novel base station flow control scheme, called Forced Duplicate ACK (FDA), has been proposed for improving the performance of wireless Internet access. In FDA, each BS monitors its average buffer occupancy for detecting incipient congestion. When a pre-defined threshold is exceeded, the BS generates three forced duplicate ACKs to a set of selected senders. Upon receiving the forced duplicate ACKs, a sender slows down its transfer rate to prevent the BS congestion. To combat the problem of multiple packet dropping at the BS, a quality guaranteed cache release function was designed without damaging the advantages of using Snoop or New Snoop. Based on the extensive simulation results, we found that FDA can effectively help Snoop and New Snoop protocols to avoid BS congestion, and can fairly allocate the wireless link bandwidth among all TCP connections.

References:

[1] H. Balakrishnan, "An Implementation of TCP Selective Acknowledgments,"ftp://daedalus.cs.Berkeley.edu/pub/tcpsack/,1996.

[2] J.H. Hu, K. L. Yeung, et al, "Hierarchical Cache Design for Enhancing TCP over Heterogeneous Networks with Wired and Wireless Links," *GLOBECOM 2000*, Nov. 2000.

[3] H. Balakrishnan, S. Seshan and R.H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," *ACM Wireless Networks*, 1(4), Dec. 1995.

[4] E. Ayanoglu, S. Paul, et.al, "A Link-Layer Protocol for Wireless Networks," *ACM/Baltzer Wireless Networks Journal*, 1:47--60, February 1995.

[5] B. Bakshi, et.al, "Improving performance of TCP over wireless networks," *Texas A&M University Technical Report* TR-96-014, May 1996.

[6] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and Sack TCP," *Computer Commun. Review*, 1996.

[7] Thomas R. Henderson, et.al., "On Improving The Fairness of TCP Congestion Avoidance," *GLOBECOM 1998*.

[8] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE Trans. on networking*, Vol. 1, No. 4, August, 1983.

[9] Carlos M. Pazos, et.al., "Using Back-Pressure to Improve TCP Performance with Many Flows," *Infocom'99*.

[10] Masatoshi Kawarasaki, et.al., "Dynamics of TCP Flow Control over High-Speed ATM Networks," *ICC'98*.

[11] S.J. Golestani and S. Bhattacharyya, "A class of end-to-end congestion control algorithms for the Internet Network Protocols," Proc. of Sixth Intern. Conf. on Network Protocols, pp. 137-150, 1998.

[12] S.A. Jsshua, et.al., "Improving TCP Performance over Multi-Slot GSM", *ICC'99*, 1999.

[13] W.R. Stevens, "TCP/IP Illustrated," Vol. 1, Addison Wesley, 1994.

[14] V. Jacobson, "Congestion avoidance and Control," *Proc. of SIGCOMM' 88*, pp. 314-329, Aug. 1988.