

## PAPER

# APB: An Adaptive Playback Buffer Scheme for Wireless Streaming Media\*

Wanqing TU<sup>†(a)</sup> and Weijia JIA<sup>†</sup>, Nonmembers

**SUMMARY** The wireless streaming media communications are fragile to the *delay jitter* because the conditions and requirements vary frequently with the users' mobility. Buffering is a typical way to reduce the *delay jitter* of media packets before the playback, however, it will incur a longer *end-to-end delay*. Our motivation in this paper is to improve the balance between the elimination of *delay jitter* and the decrease of *end-to-end delay*. We propose a novel *adaptive playback buffer (APB)* based on the probing scheme. By utilizing the probing scheme, the instantaneous network situations are collected, and then the *delay margin* and the *delay jitter margin* are employed to calculate the *step length (sl)* which is used to adjust the playback buffer in each time. The adaptive adjustment to the playback buffer in APB enables the *continuous* and *real-time* representation of streaming media at the receiver. Unlike the previous studies, the novelty and contributions of the paper are: a) Accuracy: by employing the instantaneous network information, the adjustment to the playback buffer correctly reflects the current network situations and therefore achieves the improved balance between the elimination of *delay jitter* and the decrease of *end-to-end delay*; Hence, APB adjustment is accurate in terms of improving such balance; b) Efficiency: by utilizing the simple probing scheme, APB achieves the current network situations without the complex mathematic predictions, which enables the adjustment to be more timely and efficient. Performance data obtained through extensive simulations show that our APB is effective to reduce both *delay jitter* and *playback buffer delay*.

**key words:** *playback buffer, delay jitter, delay, adaptive scheme, probing scheme*

## 1. Introduction

Streaming media (e.g. audio, video and multimedia documents) are continuous and time-based that utilize the streaming transmission technologies in the Internet/Intranet. Wireless interactive streaming media applications (e.g. the wireless multimedia chats and the wireless games) which are characterized with the frequently generated video and audio communications between two users become more and more popular in recent years. In these applications, the multimedia data are transmitted to the destination through some physical media (e.g. radio). During the transmission, besides the fluctuations in network throughput, the users' mobility and the frequent variation of the conditions and requirements in wireless communications make the wireless streaming media more fragile to the *delay jitter* as compared with the wired streaming media. *Delay jitter* is the variance in one-way latency and is calculated based on sending

and receiving time stamps of consecutive packets sent out. Large *delay jitter* will cause highly objectionable disruption or stalling in playback. Hence, eliminating the *delay jitter* before the playback is necessary for the wireless streaming media.

A typical way to reduce the frequency of playback disruption is to employ a playback buffer at the receiver. The receiver delays the playback and places the streaming media data in the playback buffer before the first packet is represented. When the data in the buffer reach the predetermined threshold, known as the *playback point*, the buffered data will start to play. With such buffering, the receiver can pre-fetch data that is not immediately needed when the available bandwidth is above the media rate, which prepares for providing data to the receiver when the available bandwidth is below the media rate. A. Cai et al. [1] described such playback buffer with a fixed volume. The idea is that the playback of the  $i$ -th packet should be later than its arrival in order to reduce the frequency of unacceptable *delay jitter*. According to this, the playback buffer volume  $B$  is calculated as

$$B = 2[(d_n)_{max} - (d_n)_{min}]r \quad (1)$$

where  $(d_n)_{max}$  and  $(d_n)_{min}$  refer to the maximum and the minimum network transmission delays respectively,  $r$  is the playback rate and the expression  $\lceil \bullet \rceil$  represents the smallest integer that is greater than  $\bullet$ . In such a way, the playback disruption will not occur when the available bandwidth is temporarily below the media rate, unless the buffer is empty. This mechanism is simple. However, the additional and fixed buffering of media data incurs a longer *end-to-end delay* before the first packet is represented at the receiver. This is not good for the interactive streaming media communications because they have strong requirements for the short *end-to-end delay*. There is a trade-off between reducing the *delay jitter* and decreasing the *end-to-end delay*.

R. Ramjee et al. [2] considered this trade-off by investigating the performances of four different algorithms for adaptively adjusting the playback delay of audio packets in face of the varying network delays. The idea behind the four playback algorithms follows the *absolute timing method* as defined by Montgomery [3]. When the connections among the users are set up, any user may initiate one or more than one conversation with other users. Each such conversation is called a *talkspurt*. Hence, the interactive streaming media communication may include several talkspurts. If packet  $i$  is the first packet of the  $i$ -th talkspurt in the interactive stream-

Manuscript received April 11, 2005.

Manuscript revised May 18, 2005.

<sup>†</sup>The authors are with the Department of Computer Science, City University of Hong Kong, Hong Kong, China.

\*This paper was presented at the IEEE International Conference on Networks (ICON 2004).

a) E-mail: tu.wanqing@student.cityu.edu.hk

DOI: 10.1093/ietcom/e88-b.10.4030

ing media communication, its *playback point*  $p_i$  is computed by

$$p_i = t_i + \hat{d}_i + 4 * \hat{v}_i \quad (2)$$

where  $\hat{d}_i$  and  $\hat{v}_i$  are estimates of the mean and variation in the *end-to-end delay* of the  $i$ -th talkspurt,  $t_i$  is the time at which packet  $i$  is generated at the sender. And the *playback point*  $p_j$  for any subsequent packet  $j$  in the talkspurt in which packet  $i$  is its first packet is computed by

$$p_j = p_i + t_j - t_i \quad (3)$$

The four algorithms adopt different ways to calculate  $\hat{d}_i$ . All the ways are based on the estimates. Such estimates cannot reflect the dynamic network delays. It is important to consider the dynamic network delays in wireless communications because the conditions in wireless communications vary frequently.

D.L. Stone et al. [4] presented the *queue monitoring policy*. In [4], the *end-to-end delay* is estimated by the length of playback queue. If the queue size remains the same during a certain period, the policy considers that the *end-to-end delay* is constant; if the queue shrinks (expands), the policy considers that the *end-to-end delay* increases (decreases). W. Xu et al. [5] followed the policy of [4] and proposed the *delay balance algorithm*. The *delay balance algorithm* adjusts the playback buffer based on the prediction. It predicts the current playback buffer length  $BL(t)$  according to the previous network situations.  $BL(t)$  equals to the maximum amount of packets that have arrived at the receiver but cannot be represented at the time  $t$ . Suppose the communication begins at the time 0.  $BL(t)$  is calculated by

$$BL(t) = s(t) - R_p[t - d_n^{\sim}(t)], t > t_{d_n^{\sim}(t)} \quad (4)$$

where  $s(t)$  is the total number of packets that have arrived at the receiver during the interval  $[0, t]$ ,  $R_p$  is the media data transmission rate,  $d_n^{\sim}(t)$  is regarded as the maximum network delay at the time  $t$  by the algorithm and  $t_{d_n^{\sim}(t)}$  is the time that  $d_n^{\sim}(t)$  appeared.  $t > t_{d_n^{\sim}(t)}$  means that the playback buffer length  $BL(t)$  at the time  $t$  is achieved based on the knowledge of previous network situations. The expression  $R_p[t - d_n^{\sim}(t)]$  refers to the minimum number of packets that the receiver can represent in the interval  $[0, t]$ . The algorithm also gives the dynamic range of the playback buffer. In order to guarantee the *continuous* playback, the algorithm demands that there is no underflow in the buffer during the communications, hence, the dynamic range of the playback buffer length  $DBL(t)$  at the time  $t$  is

$$0 \leq DBL(t) \leq R_p d_n^{\sim}(t) \quad (5)$$

The prediction of the algorithm cannot deal with the ‘‘bursts’’ of network communications. Moreover, it is inaccurate to achieve  $BL(t)$  and  $DBL(t)$  by using the previous maximum network transmission delay especially when the algorithm is used in the wireless network in which the communication environments are changing all the while.

This paper proposes a novel *adaptive playback buffer*

(APB) based on the probing scheme. By utilizing the probing packets, the instantaneous network situations (i.e. the current *round trip times*) are collected. Based on such *round trip time*, the instantaneous one-way network transmission delay is achieved by APB scheme, and then the *delay margin* and the *delay jitter margin* are employed (the *delay margin* and the *delay jitter margin* will be defined in Sect. 3) to calculate the *step length* ( $sl$ ) which is an important parameter used to adjust the playback buffer in each time. The adaptive adjustment to the playback buffer in APB achieves the *continuous* and *real-time* representation of streaming media at the receiver. Unlike the previous works, the novelty and contributions of the paper are:

- *Accuracy*: by employing the instantaneous network information, the adjustment to the playback buffer correctly reflects the current network situations and therefore achieves the improved balance between reducing the *delay jitter* and decreasing the *end-to-end delay*; Hence, APB adjustment is accurate in terms of improving such balance;
- *Efficiency*: by utilizing the simple probing scheme, APB achieves the current network situations without the complex mathematic predictions; Hence, APB is more timely and efficient in response to the variations of network situations when it is used to adjust the playback buffer for short *end-to-end delay* and acceptable *delay jitter* streaming media communications.

The rest of the paper is organized as follows. In Sect. 2, we present our motivation to design APB and the criteria to evaluate the playback performances of streaming media. We then describe the *adaptive playback buffer* in Sect. 3. Section 4 evaluates the performances of APB and other playback buffers by using the computer simulations. In Sect. 5, we conclude the paper and give our future works.

## 2. Motivation and Criteria

Buffering the data at the receiver until the predetermined threshold is reached will eliminate the *delay jitter* and then the streaming media can be represented smoothly for the users. However, the additional and fixed buffering of media data incurs a longer *end-to-end delay* before the first packet is represented at the receiver. This is not good for the interactive streaming media communications because they have strong requirements for the short *end-to-end delay*. Our motivation in this paper is to improve the balance between reducing the *delay jitter* and decreasing the *end-to-end delay* to achieve the acceptable playback performances at the receiver. Generally, two criteria: *continuity* and *real time* that correspond to the acceptable *delay jitter* and the short *end-to-end delay* respectively are used to evaluate the playback performances of wireless streaming media communications.

- *Continuity*: it refers to the streaming media data that can be represented without pattern dither and audio disruption at the receiver. *Continuity* can be achieved by

reducing the *delay jitter*. The *sufficient and necessary condition* for realizing *continuity* is *All the intervals between two adjacent transmission delays are equal. Namely,*

$$DI_1 = \dots = DI_i = \dots = DI_{n_p-1}, i \in [1, n_p - 1] \quad (6)$$

where  $n_p$  is the total number of packets transmitted in the streaming media communication,  $DI_i$  is the  $i$ -th transmission delay interval (i.e. the interval between the  $i$ -th and the  $(i + 1)$ -th packets). In [6] and [16], the *delay jitter* not exceeding 10 ms is acceptable for the video streams with the compressed TV quality and the audio streams. In our experiments, we use the *delay jitter bound*  $J = 10$  ms.

- *Real Time:* the wireless interactive streaming media communications are *real-time* if the users can receive the media data within the acceptable delay bound. *Real time* can be achieved by decreasing the *end-to-end delay*. In [6], [7] and [16], the *end-to-end delay* exceeding 250 ms is unacceptable, in general, the users are satisfied with the delay performance if the *end-to-end delay* is under 200 ms for the video streams with the compressed TV quality and the audio streams. The *sufficient and necessary condition* for realizing *real time* is *The end-to-end delay of each multimedia packet is not greater than the end-to-end delay bound DB. Namely,*

$$TD_i \leq DB, i \in [1, n_p] \quad (7)$$

where  $TD_i$  is the *end-to-end delay* of the  $i$ -th packet. In our experiments, we use the *end-to-end delay bound*  $DB = 250$  ms.

To achieve the *continuous* and *real-time* playback at the receiver, an *adaptive playback buffer* is designed in the next section.

### 3. Adaptive Playback Buffer (APB)

In this section, we first provide the architecture of the *adaptive playback buffer*, then we give a novel method to adjust the playback buffer based on the instantaneous network situation, the *delay jitter bound* ( $J$ ) and the *end-to-end delay bound* ( $DB$ ).

Figure 1 shows the architecture of the *adaptive playback buffer* in the wired-cum-wireless network. *APB Controller* is the component that adaptively adjusts the playback

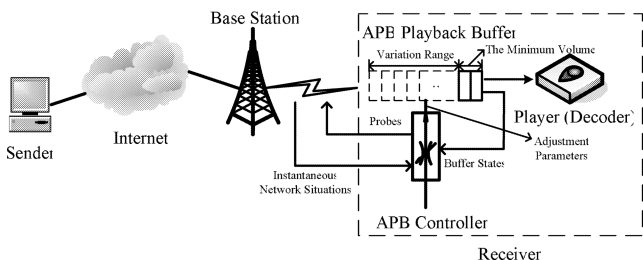


Fig. 1 The wired-cum-wireless network with APB in the receiver.

buffer. As shown in this figure, it achieves the *instantaneous network situations* (i.e. the current *round trip times*) in one of its input ports through sending the *probes* (as we will introduce in Sect. 3.1) to the packet sender in one of its output ports. Based on the current *round trip time*, *APB Controller* achieves the current one-way network transmission delay, and then calculates the current playback buffer delay. Also, *APB Controller* enquires the other input port at which the playback buffer inputs the information of its current *buffer states*. At last, *APB Controller* adjusts the playback buffer through sending the *adjustment parameters* to the playback buffer. The *adjustment parameters* are decided based on the current *buffer states* and the *step length* ( $sl$ ) that is calculated by using the current variation of playback buffer delay and the *delay jitter bound*.

To adjust the playback buffer, it is important to know that the performances not exceeding the *delay jitter bound* and the *end-to-end delay bound* should be acceptable. We can adjust APB with  $J$  and  $DB$  to improve the balance of achieving the *continuous* and *real-time* playback at the receiver. We first give the following definitions.

- *Delay jitter margin:* it is defined as the difference between the current *delay jitter* and  $J$ .
- *Delay margin:* it is defined as the difference between the current *end-to-end delay* and  $DB$ .

Our basic idea of adjusting APB is: through combining with the current playback buffer delay that is calculated by *APB Controller*, we can use the *delay jitter margin* to decrease the *end-to-end delay* and the *delay margin* to reduce the frequency of *delay jitter*.

#### 3.1 Network Transmission Delay

The one-way network transmission delay  $d_n$  is a key parameter to fluent the work of *APB Controller*. *APB Controller* utilizes the probing scheme to achieve  $d_n$ . Although the probing scheme has been studied by many well-known systems [8]–[10], it is novel to use it in the playback buffer.

Denote the time length of the probing period as  $T_p$ . Use  $n$  ( $n \in N$ ) to represent the probing period of the  $n$ -th probe sent by *APB Controller* during the interactive streaming media communication. Suppose the receiver sends the first probe at the time 0. Then,  $(n - 1)T_p$  is the time at which the  $n$ -th probe is sent out at the receiver. In this paper, we use the term “current” in APB scheme to refer to the  $n$ -th probing period. The current network transmission delay  $d_n(p, n)$  in Fig. 1 can be expressed as

$$d_n(p, n) = \sum_{i=1}^m \frac{p}{c_i(n)} + \sum_{i=1}^m \frac{q_i(n)}{c_i(n)} + d_l + \frac{p}{c'(n)} + d(n) \quad (8)$$

where  $m$  is the number of wired links between the sender and the receiver,  $p$  is the packet size,  $c_i(n)$  is the current capacity of the  $i$ -th wired link,  $q_i(n)$  is the current queueing length at the router connecting to the  $i$ -th wired link,  $d_l$  is the sum of  $m$  wired links’ propagation delays and is usually regarded as a constant because the propagation delay of each

physical link is a constant and the packet transmission between the same sender and receiver always takes the same  $m$  links in the wired network,  $c'(n)$  is the current capacity of wireless links between the base station and the mobile nodes and  $d(n)$  includes the current propagation delays and contention delays of wireless links. Unlike the wired network, the propagation delay between the base station and the mobile node is variable because the mobile environment (e.g. the location of the mobile node) is always changing.

In APB probing scheme, the *APB Controller* of each receiver periodically sends the light weight probes that carry the sending and receiving timestamps to collect *round trip time* ( $RTT$ ).  $RTT$  is one of the important network delay properties. Suppose  $RTT_{sr}(n)$  records the *round trip time* value of the  $n$ -th probe from the sender  $s$  to the receiver  $r$ . Considering the burst and fluctuation of data communications, the exponential average value of  $RTT_{sr}$ ,  $EARTT_{sr}$ , is utilized to estimate the network performance. The current  $EARTT_{sr}(n)$  is calculated by

$$EARTT_{sr}(n) = \beta(EARTT_{sr}(n-1)) + (1-\beta)RTT_{sr}(n), n \in N \quad (9)$$

where  $EARTT_{sr}(n-1)$  is the exponential average value of the *round trip time*  $RTT_{sr}(n-1)$  achieved by the  $(n-1)$ -th probe, and  $\beta$  ( $0 < \beta < 1$ ) is a smoothing factor that determines the weight given to  $EARTT_{sr}(n-1)$ . We would like to give a greater weight to more recent instance  $RTT$ . Thus, in our experiments, we use  $\beta = 0.1$ .

The current one-way network transmission delay  $d_n(p, n)$  is  $\frac{EARTT_{sr}(n)}{2}$ . We will introduce in the next subsection that, upon achieving  $d_n(p, n)$ , *APB Controller* utilizes (11) to calculate the current playback buffer delay and then adjusts the playback buffer volume according to this buffer delay.

### 3.2 Adaptive APB Size

In this subsection, we give the APB method to adaptively achieve the current APB size based on the current *delay margin* and  $d_n(p, n)$ .

Generally speaking, the *end-to-end delay*  $d_{e2e}(n)$  experienced by the multimedia packet  $p$  during the  $n$ -th probing period in Fig. 1 is

$$d_{e2e}(n) = d_n(p, n) + d_b(n) + d_c \quad (10)$$

where  $d_b(n)$  is the packet buffer delay during the  $n$ -th probing period and  $d_c$  is the codec delay. For the audio streams,  $d_c$  is a same constant in the same codec scheme for the sender and the receiver; for the video streams,  $d_c$  is a same constant for the frames with the same type (e.g. Introcoded frames in MPEG-1) by the same codec for the sender and the receiver. Hence, *APB Controller* considers that  $d_c$  is a constant after it recognizes the frame type which can be achieved through the negotiation of the codec scheme between the sender and the receiver. *APB Controller* intends to guarantee the *real time* when it adjusts the playback buffer

to reduce the *delay jitter*. The communications with the *end-to-end delay* not exceeding  $DB$  are *real-time* communications. In general, there is a difference between  $d_{e2e}(n)$  and  $DB$ . It is the current *delay margin*. The current *delay margin* provides the extra playback buffer delay  $d_b(n)$  to smooth the *delay jitter* of packets caused by the varying network situations during the packet transmission. Recall  $d_n(p, n) = \frac{EARTT_{sr}(n)}{2}$ . The playback buffer delay in the  $n$ -th probing period  $d_b(n)$  is calculated by

$$\begin{aligned} d_b(n) &= DB - d_n(p, n) - d_c \\ &= DB - \frac{EARTT_{sr}(n)}{2} - d_c \end{aligned} \quad (11)$$

Thus, the current APB size  $APBS(n)$  is

$$APBS(n) = d_b(n)r = \left( DB - \frac{EARTT_{sr}(n)}{2} - d_c \right) r \quad (12)$$

where  $r$  is the streaming media transmission rate.

The following is the analysis of the variation range of APB volume. Since  $d_c$  is some constant for both audio streams and video frames with the same type by the same codec schemes, (11) shows that  $d_b(n)$  is a monotonic decreasing function with regard to  $d_n(p, n)$ . The minimum network transmission delay is calculated by

$$\begin{aligned} (d_n(p, n))_{min} &= \min \left\{ \sum_{i=1}^m \frac{p}{c_i(n)} + \sum_{i=1}^m \frac{q_i(n)}{c_i(n)} + d_l + \frac{p}{c'(n)} + d(n) \right\} \\ &= \min \left\{ \sum_{i=1}^m \frac{p}{C_i} + \frac{p}{C'} \right\} + d_l + \widehat{d(n)} \\ &= \sum_{i=1}^m \frac{p}{C_i} + \frac{p}{C'} + d_l + \widehat{d(n)} \end{aligned} \quad (13)$$

where  $C_i$  and  $C'$  are the maximum capacities of the  $i$ -th wired link and the wireless link between the base station and the mobile node respectively,  $\widehat{d(n)}$  only includes the current propagation delay of the wireless link. Then, the maximum buffer delay  $(d_b(n))_{max}$  is achieved when  $(d_n(p, n))_{min}$  appears.

$$\begin{aligned} (d_b(n))_{max} &= \max\{DB - d_c - d_n(p, n)\} \\ &= DB - d_c - (d_n(p, n))_{min} \end{aligned} \quad (14)$$

Correspondingly, the minimum buffer delay  $(d_b(n))_{min}$  is obtained when  $(d_n(p, n))_{max}$  appears.  $(d_b(n))_{min}$  can be expressed by

$$\begin{aligned} (d_b(n))_{min} &= \min\{DB - d_c - d_n(p, n)\} \\ &= DB - d_c - (d_n(p, n))_{max} = 0 \end{aligned} \quad (15)$$

The variation range of APB volume depends on the variation range of  $d_b(n)$ . Because the streaming media data are framed at the sender by the coder, to guarantee the continuous playback, one intact frame is necessary for the decoder

at the receiver. Thus, the variation range of APB volume  $VR$  is

$$\max\{r(d_b(n))_{\min}, v\} \leq VR \leq r(d_b(n))_{\max} \quad (16)$$

where  $v$  is the maximum frame size of streaming media. (16) follows

$$v \leq VR \leq r(d_b(n))_{\max} \quad (17)$$

### 3.3 APB Adjustment

In this subsection, we introduce APB scheme that adjusts the playback buffer volume to improve the balance between the elimination of *delay jitter* and the decrease of *end-to-end delay*. To adjust the playback buffer, *step length* is utilized by *APB Controller*. *Step length* is the maximum number of buffer units that APB can increase/decrease in each time without incurring the unacceptable *delay jitter* and *end-to-end delay*. We now introduce how to calculate the *step length* ( $sl$ ). To achieve the *real-time* communications (i.e. the *end-to-end delays* not exceeding  $DB$ ), the current APB size is calculated by (12), then the current variation of APB size is

$$\Delta APBS(n) = \left\lceil \frac{APBS(n) - APBS(n-1)}{u} \right\rceil \quad (18)$$

where  $APBS(n-1)$  is the APB size in the last probing period and  $u$  is the bit number that each APB unit contains. To avoid the playback disruption, the adjustment to APB should not exceed the *delay jitter bound*  $J$  in each time. The maximum playback buffer units  $U$  that can be adjusted without causing the playback disruption is

$$U = \left\lceil \frac{J \times r}{u} \right\rceil \quad (19)$$

where  $r$  is the streaming media transmission rate. Hence, according to (18) and (19), to guarantee *real-time* and *continuous* interactive streaming media communications, the *step length*  $sl$  is

$$sl = \min\{|\Delta APBS(n)|, U\} \quad (20)$$

Generally speaking, one of the following two playback buffer states appear when the network situations vary suddenly.

- *Underflow*: The playback buffer is *underflow* if the packets received by the buffer cannot reach the *playback point* when the receiver is willing to playback the packets. The *underflow* is caused by the packet arriving rate that is lower than the packet playback rate. When the *underflow* appears, the playback disruption or stalling is caused because the receiver has no packet to represent. The *underflow* indicates the suddenly congested network situation that generates the suddenly very long  $d_n(p, n)$ .
- *Overflow*: The playback buffer is *overflow* if the pack-

ets received by the playback buffer reach the *playback point* so quickly that the receiver is unable to process them in time. The *overflow* is caused by the packet arriving rate that is greater than the packet playback rate. When the *overflow* appears, some packets have to be discarded. The *overflow* indicates the suddenly light network traffic load that makes  $d_n(p, n)$  become very short suddenly.

Both *underflow* and *overflow* cause the disruption or stalling in playback. APB employs the following adjustment scheme to achieve the *continuous* and *real-time* playback. When *APB Controller* in Fig. 1 detects the *underflow* of playback buffer, considering both *real time* and *continuity*, it decreases the playback buffer volume quickly to the current number of packets in the buffer. This operation enables the buffered packets to be out of the buffer and represented at the receiver as soon as possible. After that, in order to smooth out the fluctuation of  $d_n(p, n)$ , *APB Controller* calculates  $\Delta APBS(n)$  by (18) to see whether APB volume can be increased or not. If  $\Delta APBS(n) > 0$ , *APB Controller* calculates the  $sl$  according to the achieved network parameter  $\frac{EARTT_{sr}(n)}{2}$  and the *delay jitter bound*  $J$  by using (18), (19) and (20) and then increases the buffer volume with one  $sl$  in each time. Otherwise, *APB Controller* always adjusts the buffer volume to the current number of packets in the buffer.

When *APB Controller* detects the *overflow* of playback buffer, in order to avoid the packet loss, it increases the playback buffer volume quickly to its maximum volume  $r(d_b(n))_{\max}$  so as to contain many more received packets. After that, considering the *real-time* communication, *APB Controller* calculates  $\Delta APBS(n)$  in (18) to see whether the APB volume can be decreased or not. If  $\Delta APBS(n) < 0$ , *APB Controller* decreases the buffer volume by one  $sl$  calculated by using (18), (19) and (20) in each time. Otherwise, the playback buffer keeps the maximum volume to avoid packet loss because large packet loss will also cause the disruption or stalling in playback.

If *APB Controller* detects that there is no *overflow* and *underflow* in the playback buffer within an interval  $T$ , it considers that the network situation is stable. In such case, the fluctuation of *end-to-end delay* is very little. Considering of the *real-time* communication, it is unnecessary to buffer the packets as long as before. *APB Controller* decreases one  $sl$  of playback buffer units in each time to short the packet *end-to-end delay*.

## 4. Experimental Evaluation

### 4.1 Simulation Model

In this section, we use the computer simulations to compare the performances of APB with the fixed volume buffer (FVB) in [1], the delay balance algorithm (DBA) in [5], the minimum volume buffer (MINB) whose volume is the lower bound  $v$  of APB.

We use *ns-2* [15] to run our simulations on a group of

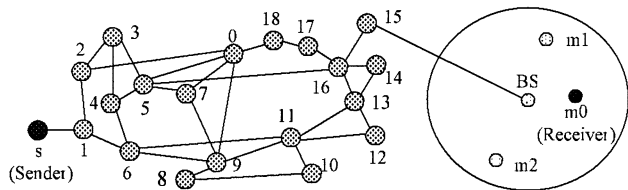


Fig. 2 Simulation topology.

SUN SOLARIS workstations. According to the wired-cum-wireless network in Fig. 1, the simulation topology adopts the integrated network shown in Fig. 2. In the topology, the backbone of the wired network is the well-known MCI ISP backbone that is a representative backbone model of Internet used in many researches and has 19 routers (the shaded nodes 0 ~ 18). The traffic sender  $s$  directly connects to router 1 in the backbone network as shown in Fig. 2. Other routers are attached with other end hosts that are not shown in the figure for the brief presentation. The bandwidth of the wired links is 100Mb. The wireless network includes 3 mobile nodes ( $m_0 \sim m_2$ ) and 1 base station (BS).  $m_0$  is the traffic receiver and equipped with one of the four playback buffers respectively to observe the playback performances of real-time streaming media communications from  $s$  to  $m_0$ . The wireless communication in the simulations adopts 802.11b (i.e. the signal transmission between the base station and the mobile nodes in the physical and data link layers follows the standard of 802.11b) that is the IEEE standard for Wireless Local Area Network (WLAN) and popular in the market today. The transmission range of base station is with the radius of 250 m that is the default setting of  $ns-2$  used in a large number of wireless researches. Because the wireless signals attenuate with the increment of the transmission distances, all mobile nodes move within the transmission range to guarantee that they can receive the clear and correct signals from BS. The simulations last 100 s. And, we simulate 10 times by using  $ns-random 0$  for each performance data which is the average result of 10 data directly generated by the simulations.

The simulations are implemented with the video streams and audio streams separately. We observe the *end-to-end delay* and *delay jitter* performances of the four playback buffer schemes under the varying network congestion. In our simulations, the video and audio streams are all generated by using the  $ns-2$  class *Application/Traffic/Trace*. The video streams are 64 Kbit/s for MPEG-4 [9]; the audio streams are 5.3 Kbit/s for G.723.1 [17]. We generate the varying network congestion by inputting different numbers of disturbance flows into the network. Table 1 lists the volumes of the four playback buffer schemes in our simulations. The simulations calculate the listed volumes by (1), (16) and (5) for FVB, APB and DBA respectively. For DBA, (5) gives the dynamic range of playback buffer length  $DBL(t)$  at the time  $t$ . The upper bounds of DBA buffer length are the maximum values of all  $R_p d_n(t)$  achieved in the video and the audio communications respectively by the

Table 1 Playback buffer volumes for the four playback buffer schemes.

	FVB(Kb)	APB (Kb)	DBA (Kb)	MINB (Kb)
Video Streams	39.375	3.28	0	3.28
Audio Streams	34.45	~ 18.13	~ 21.33	1.64
		~ 16.48	~ 18.05	

simulations. The buffer volume of MINB is the lower bound of APB that is the maximum frame size of the video and the audio streams respectively.

As for the simulations of DBA, (4) and (5) are employed to calculate the current buffer length  $BL(t)$  and the current dynamic range of buffer length  $DBL(t)$ . The calculation of these two formulas is dependent on the prediction. DBA considers the maximum network transmission delay that appears before the current time  $t$  as the current maximum network transmission delay  $d_n(t)$ , and then achieves  $BL(t)$  based on (4) to adjust the playback buffer in each time.

As for the simulations of APB, the probing scheme is implemented by generating a new  $ns-2$  agent (*Agent/Ping*) at each receiver. For the light probing traffic introduced into the network, *Agent/Ping* sends 10-byte light weight probes to the sender and waits for an acknowledgement of each probe sent. The 10-byte probing packet contains the following fields: the *sender identifier* field with 1 byte, the *receiver identifier* field with 1 byte, the *sequence number* field with 2 bytes, the *sending time* field with 3 bytes and the *receiving time* field with 3 bytes. In the video communications, the probing rate is 8 Kbit/s; and in the audio communications, the probing rate is 2 Kbit/s. To calculate  $EARTT_{sr}(n)$  in (9), we would like to give a greater weight to more recent instance  $RTT_{sr}$ . Thus, in our simulations, we use  $\beta = 0.1$ . For the *step length*  $sl$ , according to APB adjustment scheme in Sect. 3.3, the simulation programs calculate it in each time by the simple formulas developed in Sects. 3.2 and 3.3 based on the simulation traffic and the achieved  $EARTT_{sr}(n)$ .

## 4.2 Simulation Metrics

The *continuity* and *real time* are two criteria to evaluate the playback performances of interactive streaming media applications. A trade-off exists when trying to achieve both of them in the current playback buffer schemes. The balance of the *continuous* and *real-time* playback performance is expected. APB is accurate in terms of such balance and efficient to gain such balance in face of the dynamic network situations. The maximum balance between the *continuity* and *real time* shows the most accurate and efficient adjustment. In our simulations, we employ the following two metrics to evaluate the *continuity* and *real time* performances respectively, and then observe the balances between achieving the *continuous* and *real-time* communications of the four playback buffer schemes.

- Average Playback Delay: *playback delay* is defined as the *end-to-end delay* of the first packet, i.e., the delay from the first packet being sent out at the sender to the

first packet being represented at the receiver. *Playback delay* includes the codec delay, the network delay and the buffer delay of the first packet. In the simulations, we use the *average playback delay* to evaluate the *real time* of wireless streaming media communications.

- Average Playback Delay Jitter: *playback delay jitter* is defined below

$$DJ(i) = d(i) - d(i - 1) \tag{21}$$

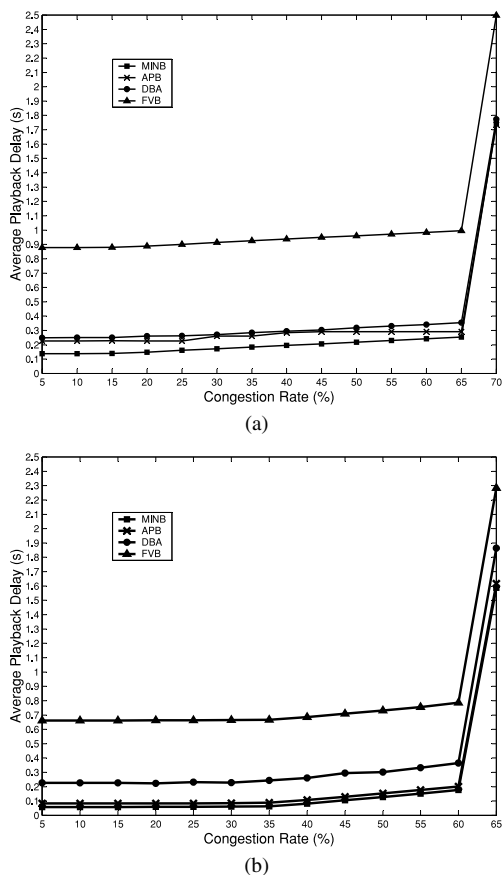
where  $d(i)$  is the current *end-to-end delay*,  $d(i-1)$  is the last *end-to-end delay*. We utilize the *average playback delay jitter* to evaluate the *continuity* of the communications.

And, in our simulations, we also observe the performances of *number of packet loss* that is an important QoS for wireless streaming media communications.

- Number of Packet Loss: it is defined as the number of packets that have been dropped before they should be represented at the receiver. Large packet loss will also incur the disruption or stalling in playback.

### 4.3 Performance Observations

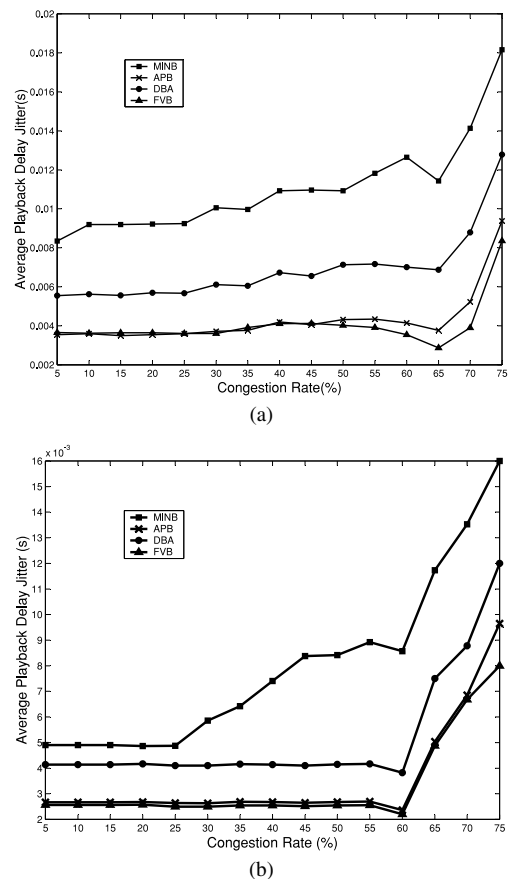
Figure 3 shows the *average playback delay* performances of



**Fig. 3** The performances of *average playback delay* under different network congestion rates for (a) video streams and (b) audio streams.

different buffer schemes for video and audio streams. The curves in Fig. 3(a) are achieved by employing video streams in the simulation. In Fig. 3(a), the *average playback delay* of MINB is always the lowest one. It is because MINB incurs the minimum buffer delay. APB achieves a shorter *average playback delay* performance than DBA and FVB. Since the communication with the *end-to-end delay* under DB (according to [6], [7] and [16], we use  $DB = 250$  ms in our simulations) is *real-time*, the simulation data in Fig. 3(a) show that the video communications are *real-time* when the congestion rate is not above 65% in MINB and APB. For DBA, the performance with the congestion rate not above 50% is acceptable. The video communications with FVB are not *real-time* for its longer buffer delay. The curves in Fig. 3(b) are achieved by employing audio streams in the simulation. Due to the same reasons above, the diagram shows the similar performance comparison as Fig. 3(a). And, Fig. 3(b) shows that the audio communications are *real-time* when the congestion rate is not above 60% in MINB and APB. For DBA, the performance with the congestion rate not above 50% is acceptable. FVB cannot achieve *real-time* audio communications at all time.

Figure 4 illustrates the *average playback delay jitter* performances of different buffer schemes for video and audio streams. The curves in Fig. 4(a) are achieved by employ-



**Fig. 4** The performances of *average playback delay jitter* under different network congestion rates for (a) video streams and (b) audio streams.

ing video streams in the simulation. In Fig. 4(a), the *average playback delay jitter* of FVB is always the lowest one. It is because FVB buffers enough packets to avoid the delay fluctuation incurred by the network variation. APB achieves a much lower *average playback delay jitter* performance than DBA. This is because the adjustment of playback buffer in APB is based on the instantaneous network situations, however, DBA is based on the prediction of the current buffer delay which cannot deal with the “bursts” of the network. MINB cannot achieve continuous communications because its *average playback delay jitter* is larger than  $J$  (according to [6] and [16], we use  $J = 10$  ms in our simulations) in most situations. The simulation data show that the *average playback delay jitter* is acceptable when the congestion rate is not above 75% with FVB and APB. For DBA, the *delay jitter* performance with the congestion rate not above 70% is acceptable. The *delay jitter* performance of MINB is unacceptable at most of the time. The curves in Fig. 4(b) are achieved by employing audio streams in the simulation. Due to the same reasons above, Fig. 4(b) achieves the similar performance comparison as Fig. 4(a). The simulation results show that the *average playback delay jitter* is acceptable in the audio communications when the congestion rate is not above 75% with FVB and APB. For DBA, the *delay jitter* performance with the congestion rate not above 70% is acceptable. MINB cannot achieve continuous audio communications when the congestion rate is above 60% in this simulation.

Table 2 (Table 3) gives the comparison of frequency that the *delay jitter* exceeds the *delay jitter bound J* under different network congestion rates for these four playback buffer schemes with video (audio) streams. The data show that APB is better than DBA and a little worse than FVB in terms of the frequency of unacceptable *delay jitter*.

Figure 5 gives the performance comparison along the *number of packet loss* of the four playback buffer schemes with the video streams (Fig. 5(a)) and the audio streams (Fig. 5(b)). In Fig. 5(a), the *number of packet loss* is increased with the increment of network congestion rate for each of the four playback buffer schemes. The curves in this

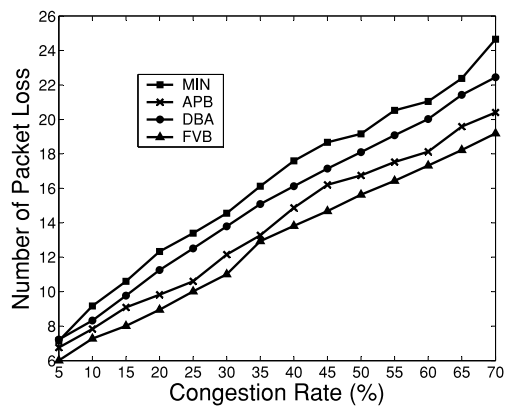
**Table 2** The comparison of frequency that the *delay jitter* exceeds the *delay jitter bound J* with video streams.

Congestion Rate	FVB	APB	DBA	MINB
5%	5	5	9	13
10%	5	5	9	15
15%	5	5	9	15
20%	5	7	11	15
25%	5	7	11	16
30%	5	7	13	27
35%	5	7	13	27
40%	6	7	14	30
45%	6	8	13	30
50%	6	8	14	29
55%	6	8	14	31
60%	6	8	14	32
65%	7	8	14	33
70%	15	16	17	32

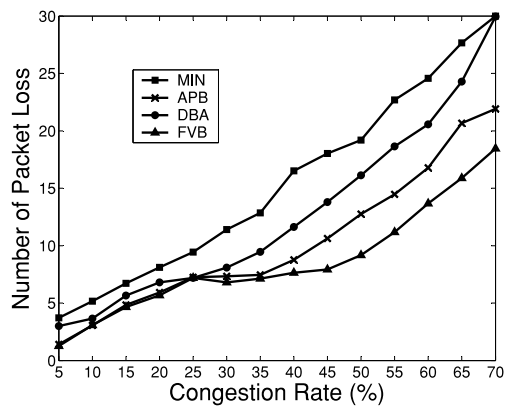
figure show that FVB achieves the best *number of packet loss* performance because its large volume may contain more received packets that cannot be represented in time. The small buffer volume of MINB causes the worst *number of packet loss* performance. It also can be seen from this figure that the packets dropped before the representation in APB is less than the ones in DBA. It is because APB utilizes the instantaneous network situations to adjust the playback buffer volume. The instantaneous network situations decide

**Table 3** The comparison of frequency that the *delay jitter* exceeds the *delay jitter bound J* with audio streams.

Congestion Rate	FVB	APB	DBA	MINB
5%	1	2	7	13
10%	2	2	7	15
15%	3	3	8	17
20%	3	3	9	16
25%	3	4	10	18
30%	3	4	12	20
35%	4	5	13	21
40%	4	6	13	23
45%	5	5	13	28
50%	5	6	13	30
55%	5	6	14	31
60%	6	6	13	33
65%	7	8	15	33
70%	14	16	18	35



(a)



(b)

**Fig. 5** The performances of packet loss under different network congestion rates for (a) video streams and (b) audio streams.

the number of packets arriving at the receiver. Thus, APB adjustment follows the variation of the number of packets arriving at the receiver. Large packet loss will also incur disruption or stalling in playback. The curves in Fig. 5(b) are achieved by employing the audio streams in the simulation. Due to the same reasons above, the diagram shows the similar performance comparison as Fig. 5(a). Hence, in the view of *number of packet loss*, the playback with FVB is more *continuous* than others and APB is the next one to FVB.

In summary, the simulation results show that the communications with APB are *continuous* and *real-time* when the congestion rate is not above 65% for the video streams and 60% for the audio streams. That is, the *balance points* of APB in our simulations are 0.65 for the video streams and 0.6 for the audio streams. However, the communications with DBA are acceptable when the congestion rate is not above 50% for both video streams and audio streams. That is, the *balance points* of DBA in our simulations are all 0.5 for both video streams and audio streams. It is because of the accurate and efficient adjustment based on the instantaneous network situations to the playback buffer in APB. As for FVB, although it can smooth out the delay fluctuation, its delay is not acceptable for the users. MINB has a good delay performance, but it causes the objectionable disruption or stalling in playback. Through the above analysis, in these four schemes, APB is the most accurate and efficient one because it achieves the best balance between reducing the *delay jitter* and decreasing the *end-to-end delay* by employing the instantaneous network situations to adaptively adjust the playback buffer volume.

## 5. Conclusions and Future Works

In this paper, we have studied the *adaptive playback buffer* (APB) to guarantee the *real-time* and *continuous* wireless streaming media communications in a compensatory way. We proposed a novel scheme—sending probing packets to achieve the instantaneous network situations to adaptively adjust the playback buffer with the *delay margin* and the *delay jitter margin*. The adjustment to the playback buffer is effective in the sense that it can solve the trade-off between reducing the *delay jitter* and decreasing the *end-to-end delay*. The simulation results show that the communications with APB are *continuous* and *real-time* when the network congestion rate is not above 65% for the video streams and 60% for the audio streams; for DBA, its performances are acceptable when the congestion rate is not above 50% for both video streams and audio streams; for FVB, although it can smooth out the delay fluctuation well, its delay is not acceptable for the users; for MINB, it has a good delay performance, but it causes the objectionable disruption or stalling in the playback. Hence, as compared with other schemes, our *adaptive playback buffer* can achieve the best balance in guaranteeing the *real-time* and *continuous* wireless interactive streaming media communications indeed.

This paper focused on the problem of improving the

balance of the two playback performances: *continuity* and *real time* by utilizing the probing scheme that accurately and efficiently reflects the instantaneous network situations. The work presented in this paper is part of our effort to design the solutions that can guarantee the acceptable playback performances at the receivers in the wireless interactive streaming media applications. There are many probing techniques, such as packet-pair probing [20], [21], packet-train probing [22], [23], Bayesian probing [24] and Markovian probing [25]. Different probing techniques and different probing rates will influence the adjustment performances of APB. For instance, a higher probing rate implies a “faster” reaction of APB to the network situations. However, a higher probing rate incurs more probe traffic which is not desirable. In this paper, we only studied the improvements in the balance of *continuity* and *real time* by using APB. We will continue the topic on the *optimal* probing technique and probing rate in terms of the accurate and efficient adjustment in our future studies.

## Acknowledgments

The authors are grateful to the Editor and the Reviewers for their precious and valuable comments to direct us to revise the paper.

This work is also partially sponsored by City University of Hong Kong strategic grants 7001709, 7001587 and 7001777 and the National Basic Research Program (973) MOST of China under Grant No. 2003CB317003.

## References

- [1] A. Cai and J. Shuen, *Multimedia Communications*, Electronics and Industry Publishing Company, 2000.
- [2] R. Ramjee, J. Kurose, D. Towsley, and H. Schularinne, “Adaptive playout mechanisms for packetized audio applications in wide-area networks,” *Proc. IEEE INFOCOM*, vol.2, pp.680–688, Toronto, Canada, June 1994.
- [3] W. Montgomery, “Techniques for packet voice synchronization,” *IEEE J. Sel. Areas Commun.*, vol.SAC-6, no.1, pp.1022–1028, Dec. 1983.
- [4] D. Stone and K. Jeffay, “An empirical study of delay jitter management policies,” *Multimedia Syst.*, vol.2, no.6, pp.267–279, Jan. 1995.
- [5] W. Xu, L. Tang, and Zh. Tan, “Design and implementation of delay balance algorithm for mutual audio in the local Ethernet,” 7-th International Conference on Multimedia Technology, pp.133–137, Oct. 1998.
- [6] Q. Zheng and R. Li, “Performance parameter calculation in the distributed multimedia synchronization,” *J. China Inst. Commun.*, vol.2, pp.53–57, Oct. 1999.
- [7] ITU-T G.114, *One-way Transmission Time*, May 2000.
- [8] E.W. Zegura, M.H. Ammar, Z. Fei, and S. Bhattacharjee, “Application-layer anycasting: A server selection architecture and use in a replicated Web service,” *IEEE/ACM Trans. Netw. (TON)*, vol.8, no.4, pp.455–466, Aug. 2000.
- [9] L. Breslau, E. Knightly, S. Schenker, I. Stoica, and H. Zhang, “Endpoint admission control: Architecture issues and performance,” *Proc. SIGCOM 2000*, pp.57–69, 2000.
- [10] W. Jia, D. Xuan, W. Tu, L. Lin, and W. Zhao, “Distributed admission control for anycast flows,” *IEEE Trans. Parallel Distrib. Syst.*, vol.15, no.6, pp.673–686, June 2004.

- [11] W. Tu and H. Qiu, "Research on guarantee technology of video stream QoS in the sink over IP network based on RTP," *ACTA Scientiarum Naturalium Universitatis Sunyatseni*, vol.42, no.5, pp.28–31, Sept. 2003.
- [12] W. Tu and W. Jia, "Adaptive playback buffer for wireless streaming media," *Proc. ICON 2004*, pp.191–195, Nov. 2004.
- [13] M. Zhang, "Multimedia communications and relative technologies," 7-th International Conference on Multimedia Technology, pp.53–57, Oct. 1998.
- [14] I. Busse, B. Deffner, and H. Schulzrinne, "Dynamic QoS control of multimedia applications based on RTP," *Comput. Commun.*, vol.19, pp.49–58, 1996.
- [15] UC Berkeley, LBL, USC/ISI, and Xerox PARC, ns Notes and Documentation, Oct. 1999.
- [16] W. Cai, *Multimedia Communication Technology*, p.7, The Publishing House of Xian Electronics Technology University, 2000.
- [17] Recommendation G.723.1, "Speech coders: Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s," ITU, March 1996.
- [18] M. Karam and F. Tobagi, "Analysis of the delay and jitter of voice traffic over the Internet," *IEEE Infocom*, Anchorage, AK, April 2001.
- [19] Moving Picture Experts Group, <http://www.chiariglione.org/mpeg>
- [20] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE J. Sel. Areas Commun.*, vol.21, no.6, pp.879–894, Aug. 2003.
- [21] J.C. Bolot, "End-to-end packet delay and loss behavior in the Internet," *Proc. ACM SIGCOMM*, pp.289–298, Sept. 1993.
- [22] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," *Proc. ACM SIGCOMM*, pp.295–308, Pittsburgh, PA, Aug. 2002.
- [23] R.L. Carter and M.E. Crovella, "Measuring bottleneck link speed in packet-switched networks," *Perform. Eval.*, vol.27, no.28, pp.297–318, 1996.
- [24] K. Harfoush, A. Bestavros, and J. Byers, "Measuring bottleneck bandwidth of targeted path segments," BUCS-2001-016, July 2001.
- [25] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *Proc. ACM SIGMETRICS 2000*, pp.145–155, Santa Clara, CA, June 2000.



**Weijia Jia** received his B.Sc., M.Sc., and Ph.D. degrees, all in computer science, in 1982, 1984, and 1993 from Center South University, China and Polytechnic Faculty of Mons, Belgium. He joined the German National Research Center for Information Science (GMD) in St. Augustin in 1993 as a research fellow. In 1995, he joined the Department of Computer Science, City University of Hong Kong (CityU) and, currently, he is an associate professor of the Department of Computer Science. His research interests include computer network, distributed systems, multicast and any-cast QoS routing protocols for Internet and wireless communications. In these fields, he has published more than 200 papers and books/chapters in the international journals and conference proceedings. He has been the Principal-Investigator of 18 research projects supported by RGC Research Grants, Hong Kong and Strategic Research Grants, CityU. He has served as PC co-chairs and PC members for various IEEE international conferences. He is a member of the IEEE.



**Wanqing Tu** received the B.E.E. degree in electronics and information systems from Nanchang University, Nanchang, China, in 1998, and the M.Sc. degree in computing networks from Zhongshan University, Guangzhou, China, in 2002, respectively. She is currently a Ph.D. candidate in the Department of Computer Science, City University of Hong Kong. Her research interests include QoS, overlay networks, end host multicast, distributed computing and wireless computing. She is a student member

of IEEE Communications Society.