A Cost-Effective Rate Control for Streaming Video for Wireless Portable Devices

Youn-Sik Hong[1] and Heemin Park[2]
[1] Department of Computer Science and Engineering, University of Incheon
Incheon 406-772, South Korea
[e-mail: yshong@incheon.ac.kr]
[2] Department of Multimedia Science, Sookmyung Women's University
Seoul 120-742, South Korea
[e-mail: heemin@sookmyung.ac.kr]
*Corresponding author: Heemin Park

Received February 19, 2011; revised April 11, 2011; revised May 8, 2011; accepted June 1, 2011; published June 28, 2011

Abstract

We present a simple and cost effective rate control scheme for streaming video over a wireless channel by using the information of mobile devices' buffer level. To prevent buffer fullness and emptiness at receivers, the server should be able to adjust sending rate according to receivers' buffer status. We propose methods to adjust sending rate based on the buffer level and discrete derivative of the buffer occupancy. To be compatible with existing network protocols, we provide methods to adjust sending rate by changing the inter-packet delay (IPD) at the server side. At every round-trip time, adjustments of sending rate are made in order to achieve responsiveness to sudden changes of buffer availabilities. A series of simulations and the prototype system showed that the proposed methods did not cause buffer overflows and it can maintain smoother rate control and react to bandwidth changes promptly.

Keywords: TFRC, video streaming, PDA, receiver-based rate control, roundtrip time (RTT), inter-packet delay (IPD)

A preliminary version of this paper appeared in the IEEE Systems, Man and Cybernetics (SMC), 2009 [15]. This version includes an improved method considering changes of buffer level ($\Delta B(t)$) and further explanations about the proposed methods. Also, this version presents implementation results on a prototype system of PDA devices. This research was supported by both of the University of Incheon Research Grant 2010 and the Sookmyung Women's University Research Grants 2010.

DOI: 10.3837/tiis.2011.06.004

1. Introduction

With the recent advent of form factor portable wireless devices, delivering video streams over wireless channel to mobile devices is becoming an important research topic of growing interest. As usage of mobile multimedia increases, it is expected that video streams will be a major source of communication traffic to these mobile devices. In typical VoD (Video on Demand) applications, mobile devices—in particular, PDA (Personal Digital Assistant) and smart phones—are mainly used for receiving video streams from their stationary servers. A mobile device as a receiver needs some buffer space to smooth out fluctuations of receiving data rate. At the start of a session or after fast forwards, the buffer needs to be filled up first before the video can be played. This time to fill up the buffer is the startup delay which depends on the buffer size at the mobile devices and the bandwidth available from the network. The buffer should be properly dimensioned so that the video play is continuous without underflows at the client's buffer due to the reduction of available bandwidth by network congestion. Or, over-sized buffers would result in expensive and long startup latency in client systems.

To sustain seamless video play at client devices, continuous monitoring of network status is required to prevent transmission collapse. To address this problem and to guarantee real-time streaming, some intelligent methods for rate adaptation or rate control mechanisms should be employed. One rate control scheme popularly used for streaming data in wired networks is the equation-based one, also known as TCP-friendly rate control (TFRC) [1]. In TFRC, the transmission rate is determined by a function of packet loss rate, round-trip time (RTT), and packet size; this aims to achieve a long-term and steady performance of TCP. One of the advantages of using TFRC is that it generates only small amount of rate fluctuations, which is attractive for streaming applications that require constant video quality. This TFRC has worked well for wired networks, but it is not the case for wireless networks because it is very hard to distinguish between packet losses due to transmission failures and network congestions.

A number of previous efforts have improved the performance of TFRC over wireless channel, but these have also had two main problems. First, mobile devices are normally small and thus resource-constrained with limited computing power and small buffer capacities. When a mobile client receives video stream from its stationary server, the server and the mobile client act as a fast sender and a slow receiver, respectively, because of the large gap between their transmissions capabilities. Therefore, even if there is no congestion and the bit error rate (BER) is very low in the wireless channel, the mobile client may not receive the video stream as fast as the sending rate. The second problem is that the status of mobile devices can only be determined by their buffer occupancy level (hereafter referred to as buffer level). For example, if the buffer level is less than a minimum fill level, this means the device is at the state of underflow and we can interpret this as the sending rate being too low. If the buffer level is higher than some maximum range, it will likely soon enter the state of overflow. In this case, it should prevent buffer overflow by ensuring that sending rates are not too aggressive.

Most previous work did not make use of the buffer status of receivers which is the most representative information to describe relationship between sending rate and receivers' state in video streaming applications. They have considered only the network status. Unlike the conventional approaches in previous work, we propose to reformulate the rate control method

of the TFRC by taking into account the buffer status of the client devices. A couple of advantages of our approach are as follows. First, except for the application layer, it does not require any other modifications on network protocols. Only the inter-packet delay at the sender is adjusted to control the sending rate. Second, as the rate control is performed at sender, it does not cause computation load at the receiver side. This is very useful and suitable for resource-constrained mobile devices. The objective of our approach is to provide a simple and cost effective scheme which offers satisfactory video streaming quality over noisy wireless channel using limited buffer in mobile devices.

The remainder of this paper is organized as follows. Previous work is discussed in Chapter 2. After that, proposed method to improve performance is presented in Chapter 3. The experimental results are shown in Chapter 4. Finally, we conclude in Chapter 5.

2. Previous Work

In this section, we briefly summarize previous work of rate control for streaming video over wired and wireless channel. One of the most popular end-to-end streaming protocol that uses equation for determining sending rate is presented in [1], and Padhye *et al.* [2] proposed the model of the steady state throughput for a TCP flow as a function of packet loss rate and round trip time as follows.

$$T = \frac{S}{rtt\sqrt{\frac{2p}{3}} + rto(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{1}$$

where T represents the sending rate, S is the packet size, rtt is the end-to-end RTT (round-trip time), rto is the retransmission time-out (RTO), and p is the end-to-end packet loss rate. They [1], [2] worked well for wired network, but in presence of wireless errors, they show degraded performance. Chen *et al.* [3] uses a more simplified model for TFRC as shown in Eq. 2, while it includes all the necessary factors that should affect the sending rate.

$$T = \frac{kS}{rtt\sqrt{p}} \tag{2}$$

where k is a constant factor. The authors applied the simple model for multiple TFRC connections to fully utilize wireless channel. Sisalem *et al.* [4] proposed LDA (Loss Delay-based Adaptation) which is a video streaming scheme based on RTP/RTCP [5]. LDA uses RR (Receiver Report) message in RTP/RTCP to extract the network parameters such as RTT, packet loss, and RTO. Then, LDA recalculates the sending rate using Eq. 1. SRTP [6] is similar to LDA in that it attempts to smooth out the variation of the sending rate due to the fluctuations in network speed. One typical problem of the abovementioned equation-based rate control approaches is the under-utilization of the wireless channel. For example, the calculated sending rate by the equation is only about 60% of the actual measured sending rate.

Zheng *et al.* [7] derived minimum buffer size for handheld devices. The minimum buffer size is proportional to the average retransmission time. One problem in this approach is that

minimum buffer size may increase abruptly in order to maintain continuity at the mobile device when channel BER increases.

Some of previous work focused on how to distinguish between packet loss caused by congestion and that caused by wireless channel error. Cen *et al.* [5] proposed and evaluated several algorithms to distinguish the two. Yang *et al.* [8] proposed a video transport protocol (VTP) based on estimation of achieved rate and loss discrimination algorithm. Krunz *et al.* [9] proposed an adaptive rate control scheme. They tried to reduce the bit rate of the transmitting video signal and increase error protection when the channel is expected to be bad or starvation of the playback buffer at the receiver is predicted. Then, the receiver can have two distinct states, stable and underflow depending on the threshold of playback buffer. Gualdi *et al.*'s work [10] is somewhat similar to our proposed approach in the sense that they performed adaptive control by maintaining the buffer occupancy at the decoder side between two given levels. However, they tried to control the playback frame rate in function of the buffer occupancy only. Papadimitriou *et al.* [11] proposed a receiver-centric congestion control mechanism. With this method, the transmission rate is controlled by adjusting IPG (inter-packet gap) which is similar to the IPD control used in our proposed method. If no congestion is detected, IPG is reduced additively; otherwise, it is increased multiplicatively. But, they did not provide typical values for practical IPGs. Our experimental results show that if IPD becomes greater than 20ms, the packet transmission time starts increasing exponentially. In other words, resulting IPDs to control the sending rate could be specified to a certain range by some methods.

3. A Receiver-Centric Rate Control

3.1 Fundamental Observations

The buffer occupancy $B(t)$ of a receiver at time t can be computed by the following equation.

$$B(t) = B(t-1) + T(t-1,t) - C(t-1,t) \tag{3}$$

where $B(t-1)$ represents buffered data at time $t-1$, $T(t-1)$ is the server sending rate at the time interval $[t-1, t]$, and the $C(t-1, t)$ is the amount of data stream consumed by receivers' playback at the time interval $[t-1, t]$. The unit of time t in this paper is second.

To get a more practical sense of the buffer occupancy in Eq. 3, we tried to estimate $B(t)$ using the results of our earlier work of which test environment is also video streaming case [16]. Measurements were performed on a test-bed in which a real prototype of an infrastructure network was built on a small scale. The prototype system is an infrastructure network that combines a wired LAN (Ethernet) with a wireless-LAN (IEEE 802.11b). In the test-bed, the receiver is a mobile device of PDA with 400MHz CPU. For simplicity but without loss of generality, we assume that all packets have same size S. In our previous work [16], the measured data consumption rate per second converges $1.56 \times S$ when the playback rate is constant and the buffer size was 16,384 bytes. The measured sending rate per second was approximately $10.7 \times S$. In addition, to start play back at the receiver, the buffer of the receiver has to have at least some specified number of packets represented as $B_{\min}$, which is the minimum fill level of the buffer (fixed). If we use the measured sending rate and consumption rate with regard to S and we set the current buffer level to $B_{\min}$ for asymptotic approach, then Eq. 3 can be rewritten as Eq. 4.

$$B(t) \approx B_{\min} + 10.7 \times t \times S - 1.56 \times t \times S \\ = B_{\min} + 9.14 \times t \times S \tag{4}$$

For streaming videos, stationary servers and PDAs usually act as fast senders and slow receivers, respectively due to large gap in their computation capabilities. Because the sending rate is higher than the data consumption rate, there are high probabilities of buffer fullness. This simple observation has motivated us to develop an adaptive control of the transmission rate of the server depending on the buffer level.

The minimum buffer level required at the client to prevent underflow can be approximated by Eq. 5.

$$B_{\min} \propto rtt \times C^{\max} \tag{5}$$

where $C^{\max} = \max_{0 \le i < N} [C(i-1, i)]$ and N is the length of the video in frames. Eq. 5 gives a reasonable approximation for estimating the minimum buffer level. Likewise, similar equations for estimating the minimum buffer level were derived in [7, 12] in which they used the fixed round trip time and expected data consumption rate. Contrastively, buffers in routers are designed to be larger than the bandwidth-delay product [13]. In **Fig. 1**, we show the comparative analysis on RTT with respect to different packet sizes in wired and wireless networks. Measurements were performed on the same test-bed used for estimating $B(t)$ in Eq. 4. This means same experimental setup is used as well. Because the network routing protocol is out of scope of this research, we used single hop network topology in which mobile devices can directly communicate with the base station and thus we can focus on rate control.

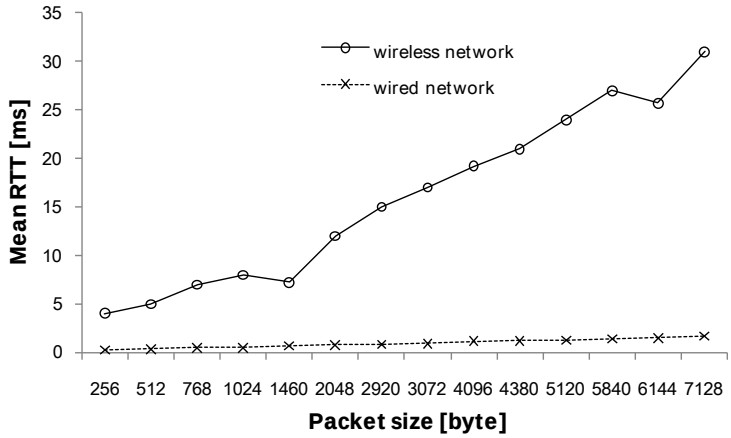

Fig. 1. Packet size versus RTT in wired and wireless networks.

As expected, although there are some fluctuations, RTT linearly increases in proportion to the packet size also in wireless networks. Assume that there is no user interaction and thus the playback rate is to be kept constant. This means the consumption rate at the receiver is constant. In addition, the minimum fill level is pre-determined by Eq. 5 and remains same during the playback. In this case, the buffer level at time t can be estimated as following equation.

$$B(t) \approx B_{\min} + T(t-1,t) \propto T(t-1,t) \tag{6}$$

As indicated in Eq. 6, the buffer level is mainly affected by the sending rate. In other words, the sending rate is also affected by the buffer level as the relationship shown in Eq. 7. This means adjustment of the sending rate can be done based on the status of the client buffer.

$$B(t) \propto T(t-1,t) \Leftrightarrow T(t-1,t) \propto B(t) \tag{7}$$

By exploiting this relationship, we propose a receiver-based rate control adjusting the server's sending rate such that the probability of both events of buffer fullness and buffer outage are minimized. Thus, the server's sending rate can be adapted by the function of the buffer occupancy at the client as shown in **Fig. 2.** In practice, if the buffer occupancy is high, the source's sending rate needs to be decreased not to cause overflow. Similarly, if the buffer occupancy is too low, the sending rate needs to be increased to prevent from running out of data in the buffer.

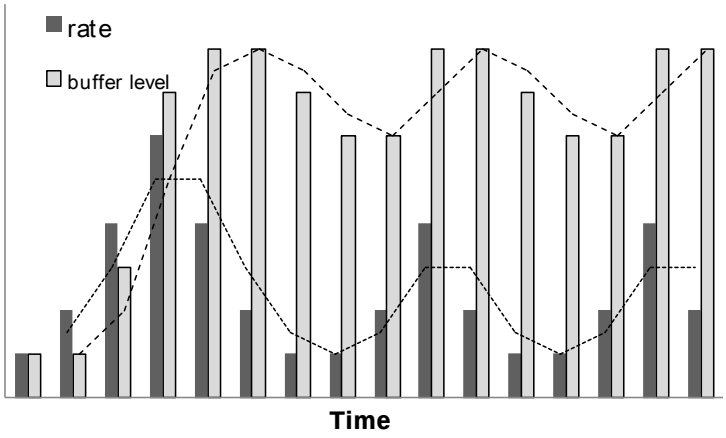

Fig. 2. Adaptation of server's sending rate to the buffer occupancy of a receiver.

3.2 Buffer States

A buffer in clients can be said to be in one of the three states: underflow, stable (normal), or overflow. Since the state of the buffer indicates the demands of buffer in the client, it is important to estimate the stationary probability of the client's buffer.

If the summation of data production rate $T(t)$ and amount of data in buffer $B(t)$ is never less than consumption rate $C(t)$, there would never be net decreases in the buffered data. This can be formulated as Eq. 8 and this is also a long-term constraint to prevent starvation which causes paused display at the client devices.

$$C(t) \le B(t) + T(t), \quad \text{where } 0 \le t \le N \tag{8}$$

If this constraint is not satisfied, the buffer will eventually enter the underflow state. In a typical case, maximum data production rate at the stationary server is higher than the consumption rate in the mobile device as shown in Eq. 4. Then, the maximum possible size of the buffer can be calculated as follows.

$$T(t) - C(t) \le B_{\max}, \text{ where } 0 \le t \le N \tag{9}$$

where $B_{\max}$ is the maximum buffer size. If the size of the buffer is close to $B_{\max}$, the buffer may enter the overflow state. This means some packets could be lost or dropped at clients. The larger the buffer size, the higher the sending rate can be supported.

We can define some safe range of minimum and enough buffer fill levels. Let B_L be the minimum buffer level and B_H be the enough buffer level for safe and stable operations. Because there is latency between feedback from the receiver and the reaction by the sender, B_H must be less than $B_{\max}$. The latency is caused by two reasons: feedback period of the receiver (i.e. RTT) and packet losses due to communication problems. The feedback latency is the time between rate change and detection of the network's reaction to that change. Because normal feedback latency is equal to one RTT in our system, we can adjust B_H using an equation of $B_H = B_{\max} - k(S/rtt), \quad k \ge 1$. The parameter k can be tuned so as to accommodate the latencies. Zheng and Atiquzzaman[7] provided an analytical model for estimating the minimum buffer level, B_L. According to [7], $B_L = T_r E[\mu(t_1)]$, where T_r is the average successful retransmission time for an error frame and $E[\mu(t_1)]$ is the mean of $\mu(t)$ which is the multimedia playback rate at time t. With a constant playback rate and a low bit error rate condition, B_L would strongly depend on the transmission rate.

Then, if the buffer level is in between B_L and B_H, the buffer can be said to be at the stable state (or normal). These three states of client's buffer can be summarized in Eq. 10.

$$\begin{aligned} \text{Underflow}: &\quad 0 \le B(t) < B_{\min} = B_L \\ \text{Stable}: &\quad B_L \le B(t) < B_H \\ \text{Overflow}: &\quad B_H = B_{\max} - \alpha < B(t) \le B_{\max} \end{aligned} \tag{10}$$

Note that α is used to introduce margin when deciding buffer overflow. It will be determined by empirically and it is usually proportional to the current sending rate. The buffer at the receiver plays an essential role in mitigating the discrepancies between data production rate (packet generation rate at the encoder side) and data consumption rate (packet extraction rate at the decoder side). The initial latency at playback time is directly related to the occupancy of the buffer. In addition, if the packet generation rate remains higher than the packet extraction rate for long period of time, the buffer will fill up after all and every packet received thereafter would be dropped.

The system needs to employ adaptive controls to achieve best trade-offs between low-latency and good video fluency; this can be done by keeping the buffer occupancy at the receiver between two specified levels B_L and B_H. In practice, two values, $B(t)$ and $\Delta B(t)$, are monitored to implement the dynamic control, where $B(t)$ is the buffer occupancy and $\Delta B(t)$ is the first-order discrete derivative of $B(t)$. For example, if the buffer occupancy is high enough and it continues increasing (i.e. $\Delta B(t) > 0$) at a certain high rate, the sending rate needs to be

reduced. Similarly, if the buffer occupancy is too low and it is still decreasing (i.e. $\Delta B(t) < 0$), the sending rate needs to be increased.

3.3 Inter-Packet Delay

In the application layer, the inter packet delay (IPD) is the time interval between transmissions of two successive packets by hosts as shown in **Fig. 3**.

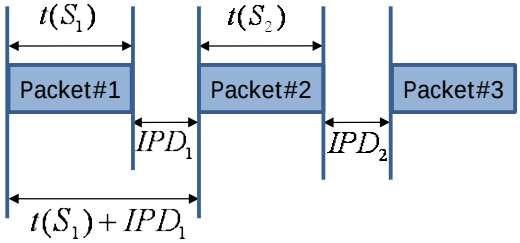

Fig. 3. A definition of inter-packet delay

The sending rate of packets can be controlled by properly adjusting IPDs. As IPD decreases, the sending rate, T, increases accordingly. Consider a source that transmits n packets of length S_1, S_2, ..., S_n during a time period [0, t], where S_i represents the i^{th} packet. Then, the average throughput achieved by the connection can be calculated by Eq. 11.

$$\begin{aligned} throughput &= \frac{1}{t}\sum_{i=1}^{n} S_i \approx \frac{1}{t} n\overline{S} \\ &= \frac{n\overline{S}}{\sum_{i=1}^{n}\left[t(S_i) + IPD_i\right]} \\ &\approx \frac{\overline{S}}{t(\overline{S}) + \overline{IPD}} \end{aligned} \tag{11}$$

where $\overline{S}$, $t(\overline{S})$, and $\overline{IPD}$ represent the average packet length, the average time to transmit them, and the average inter-packet delay, respectively. Time used to transmit i^{th} packet is $t(S_i)+IPD_i$. At every RTT period, rate adjustment is performed to provide responsiveness to the sudden changes in buffer availabilities. Considering one measurement period of RTT, flow throughput is as follows.

$$throughput \propto \left(\frac{\overline{S}}{\overline{RTT}}\right) \approx \frac{\overline{S}}{t(\overline{S}) + \overline{IPD}} \tag{12}$$

Then, the instantaneous transmission rate becomes a function of packet length (S_i) and time to send the packet [$t(S_i)+IPD_i$]. If the packet size is assumed to be constant, the function can be

approximated to have an only argument of IPD. Now, we obtain the instantaneous transmission rate T_i for the flow as follows.

$$\begin{aligned} T_i &= \frac{S_i}{t(S_i) + \overline{IPD_i}} \\ &= f\left(\frac{1}{IPD_i}, S_i\right) \\ &\approx kf\left(\frac{1}{IPD_i}\right) \end{aligned} \tag{13}$$

where k is used to adjust the resulting $IPD(t)$ such that it falls into a possible range. In **Fig. 4**, transmission time for down-streaming with respect to IPD is shown. Apparently, the increase of IPD directly affects the transmission time as well as flow throughput. In order not to degrade the overall performance, IPD should be chosen properly by considering network parameters, such as *rtt*, the size of the receive buffer, and the capacity of the link. In our test, we empirically chose IPD in the range of between 1ms and 20ms.

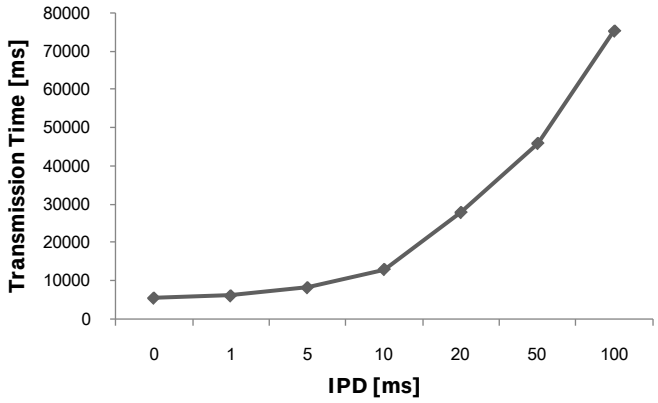

Fig. 4. The transmission time with respect to inter-packet delay during a downstream

During playback time, the state of the receiver's buffer shall be one of three states: Underflow, Stable, and Overflow State. The transitions among the three states are conducted according to the buffer occupancy $B(t)$ and its recent change $\Delta B(t)$. State transition diagram is shown in **Fig. 5**. At every state, the sending rate is adjusted with the following equations (Eq. 14, 15 and 16); the three equations show the proposed rate control schemes depending on the buffer state.

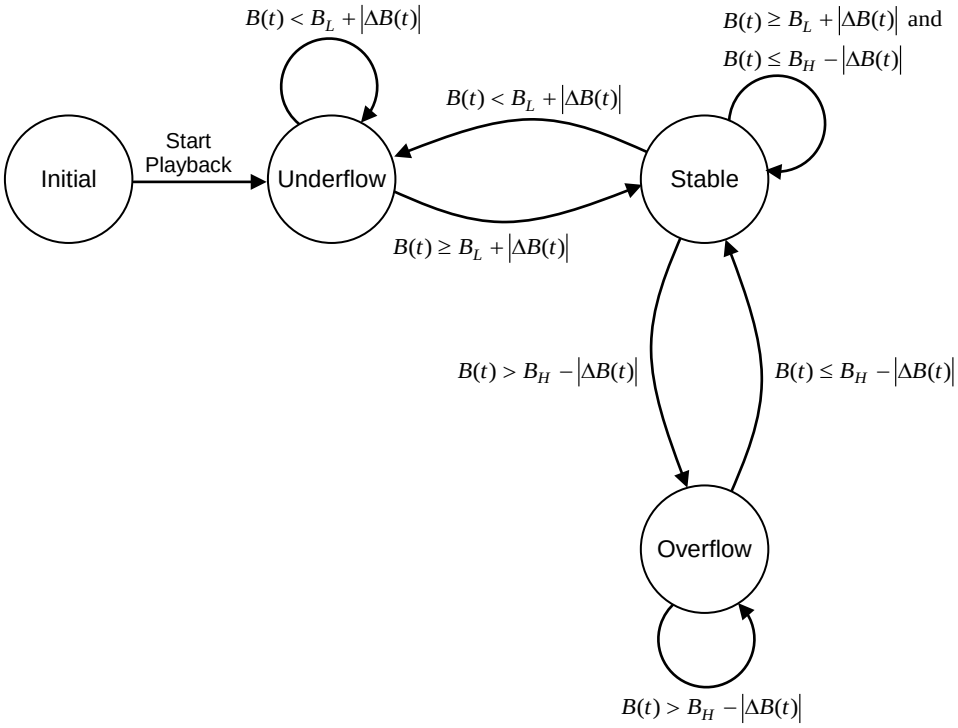

Fig. 5. State transition diagram for receiver's buffer

First, in underflow states, the sending rate, $T(t)$, should be increased as follows.

$$\text{Underflow State: } T(t) = \gamma_1 \times T(t-1), \text{ where } 1 < \gamma_1 < 2 \tag{14}$$

The parameter γ_1 is computed empirically and inversely proportional to $\Delta B(t)$. For the stable state, we applied the approach of [8] as shown in Eq. 15, where Δrtt is the amount of increase of rtt after each round.

$$\text{Stable State: } T(t) = \frac{T(t-1) \times rtt + 1}{rtt + \Delta rtt} \tag{15}$$

When the buffer enters the overflow state, the sending rate should be reduced as follows.

$$\text{Overflow State: } T(t) = \gamma_2 \times T(t-1), \text{ where } 0 < \gamma_2 < 1 \tag{16}$$

The parameter γ_2 is selected between 0 and 1 as tolerable rate reduction ratio and it is also depends on the discrete derivative of the buffer occupancy, $\Delta B(t)$.

With above three equations, we can easily implement the methods to adjust the sending rate at server side to maintain video fluency and to prevent buffer fullness and emptiness at the same time.

3.4 Feedback Interval

To make the rate control mechanism be effective, there should be frequent feedbacks from sender to the server so as to allow it to quickly react to the buffer state. In the proposed system, the receiver periodically sends feedback packets including its buffer state to the sender. Like the case of TFRC [1], the feedback should normally be sent at least once at every RTT. In practical systems, the feedback interval, i.e. RTT, is about 21-29ms as shown in **Table 1**. Because of the random nature of the RTT, using the latest RTT value as the feedback interval is likely to result in fluctuations and poor behavior. Thus, we used a smoothed version of RTT called *SRTT* that represents low frequency variations of RTT and filters out the transient changes as shown in Eq. 17.

$$SRTT(t) = \alpha \times SRTT(t-1) + (1-\alpha) \times rtt(t-1) \tag{17}$$

4. Performance Evaluation

4.1 Experimental Setup

To provide high validity of our proposed approach, we performed experiments on two types of evaluation platforms. One type of experiments was performed with the ns-2 simulator [14] to analyze the efficiency of the proposed rate control in various error rate conditions. The topology of [15] representing a mixed wired-cum-wireless network is used throughout the evaluation. The other type of experiments was done on a small-scale testbed of PDA platform which is a real working prototype. In these experiments, we traced the sending rate to see whether it quickly reacts to buffer level changes.

4.2 Estimation of Minimum and Maximum Buffer Size

First, we measured *rtt* (in average) under various error rates through ten runs and we estimated the minimum buffer level and the maximum buffer level by using Eq. 5 and Eq. 9. These estimated parameters have been used to determine the adjustable constant α in Eq. 10. **Fig. 6** shows the decrease of the congestion window according to the increase of the error rate. This means the sending rate is lowest at the highest error rate. The calculated sending rates from the predefined error rates and the measured *rtt* (in average) by using Eq. 2 are summarized in **Table 1**, where the constant k = 1,502. As indicated in Eq. 1 and Eq. 2, the achievable sending rate is inversely proportional to the error rate and RTT as well.

Note that the packet size S is kept to be constant. This simulation result shows that *rtt* has little changed within the error rate of less than 10%. In the absence of random errors, the measured *rtt* was roughly 19ms. Assume that the error rate is less than 10% and the data consumption rate is to be kept constant, $B_{\min}$ depends on the consumption rate; $B_{\min} > rtt \times C^{max} \approx rtt \times C_{\text{const}} \approx 20\text{ms} \times C_{const}$. Our empirical results gave us the following measured intuition: $C_{measured} / T_{measured} = 1.56 \times S / 10.7 \times S = 0.15 < 0.2$. Thus, the estimated lower bound on the buffer level is about 20% of the fixed buffer size.

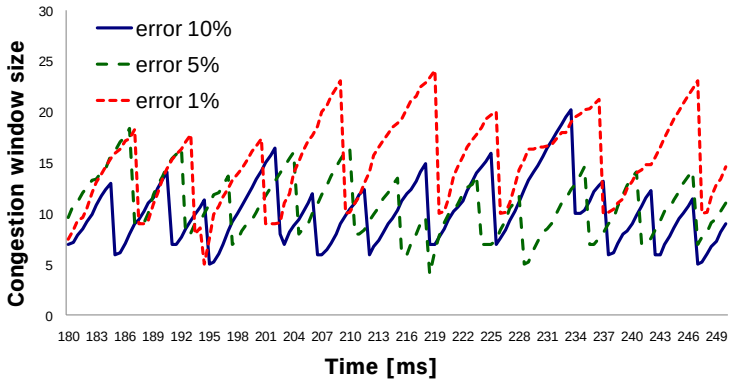

Fig. 6. The congestion window under different control error rate.

Table 1. The calculated sending rate under different random error rate.

Error Rate	5%	10%	15%	20%
rtt [ms]	21.4	22.05	23.75	29.15
Sending Rate	313.47×S	215.12×S	163.07×S	115.06×S

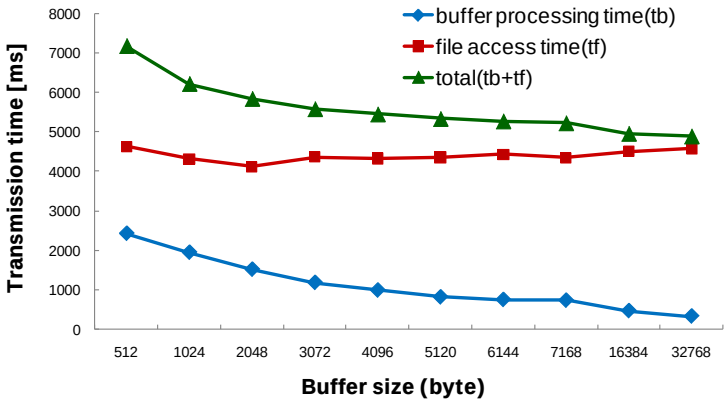

Fig. 7. The transmission time with respect to the buffer size.

In **Fig. 7**, the behavior of the transmission time with respect to the size of the receiver buffer is shown. Note that the socket buffer size of PDA is limited to 32,768 bytes. Because larger buffer size results in shorter processing time as shown in **Fig. 7**, the parameter α in Eq. 10 becomes smaller with larger buffer size. This means the data consumption rate also increases as the buffer size increases.

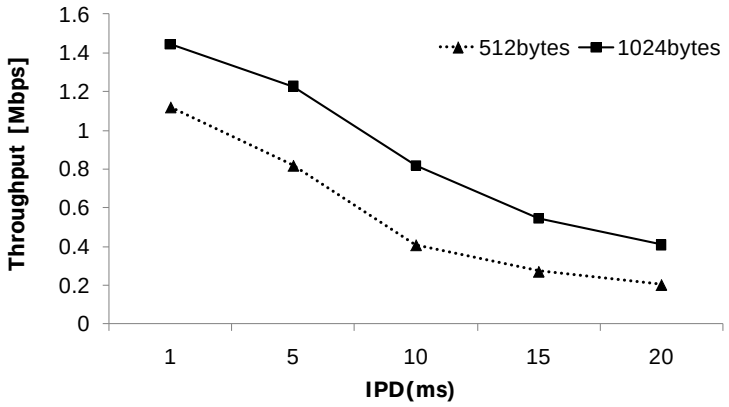

Fig. 8. The transmission time with respect to IPD.

4.3 Rate Control using Inter-Packet Delays

We tested the rate variations according to IPD changes under different packet sizes. The results of this test prove the agility of our proposed method and its efficiency. As shown in **Fig. 8**, with smaller packet size, the rate variation to IPD changes more quickly. By setting the IPD to 20ms, the rate will be drastically decreased to 24% of that of 1ms.

4.4 Adaptive Rate Control and Its Effectiveness

We conducted simulations with varying each of the parameters α, B_{min}, and B_{max}. Like our previous works [15], B_{min} and B_{max} are selected based on the least standard deviation. That is, B_{min} and B_{max} were set to 20% and 90% of the full buffer size (i.e. α =10%), respectively. In addition, typical values used in the system, were obtained via several tests: γ_1 = 1.2 in Eq. 14 and γ_2 = 0.9 in Eq. 16. The values of γ_1 and γ_2 can be slightly adjusted depending on the $\Delta B(t)$. For example, if the buffer level remains in the underflow state and continues to increase (i.e. $\Delta B(t) > 0$), γ_1 is set to 1.3 instead of 1.2. On the other hand, in the case of overflow state, if the buffer level continues to decrease (i.e. $\Delta B(t) < 0$), γ_2 is set to 0.8 instead of 0.9 for reducing the rate at a higher rate.

Fig. 9 shows the sending rate with respect to the buffer level in the case of the receiver buffer size of 16,384 bytes. When the fluctuation of the buffer level is relatively low, the sending rate changes slowly. However, as the fluctuation of the buffer level becomes high, the sending rate starts changing rapidly. As mentioned earlier, rate adjustment is performed at every RTT. So, the period of adjusting rate control is same as RTT. With the ns-2 simulation, typical value of RTT was in between 20ms and 30ms. However, in a real prototype system, the maximum RTT was 55ms. In our previous works [15], the discrete derivative of the buffer occupancy, $\Delta B(t)$, was not considered to adjust the sending rate. Let us call this system type-II method and our proposed method in this paper type-I method, respectively. **Fig. 10** shows the comparison results of the buffer occupancy of the receivers over time for both of the type-I and the type-II method.

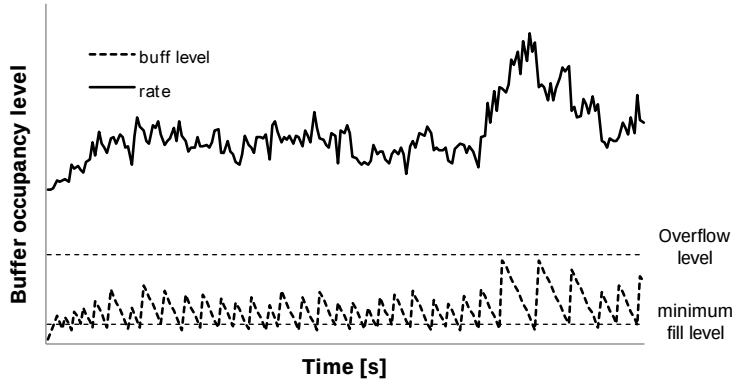

Fig. 9. The sending rate versus buffer occupancy.

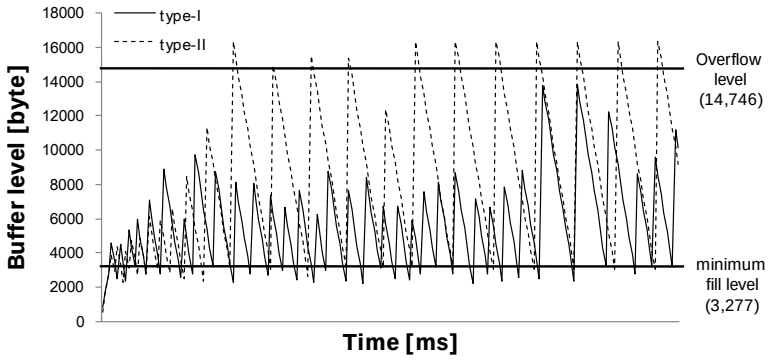

Fig. 10. The comparison results of the receiver buffer occupancy for type-I and type-II method (with the buffer size of 16,384 bytes).

With the type-I method, there were no cases of overflow state. But, the buffer of type-II method entered into overflow state eleven times during the simulation period. Since the type-I method can predict the trend of the buffer occupancy using $\Delta B(t)$, the possibility of over-utilization of the buffer can be low. When streaming video from a stationary server to a mobile station such as PDA, the server's sending rate could be faster by about 5 times than the consuming rate at the receiver. So, if the system fails to adjust the rate in timely fashion, the possibility of entering the overflow state increases, especially in case of high sending rate. In addition, if the buffer fullness happens without knowledge of status and trend of the buffer occupancy, return to a stable state by reducing the sending rate could be very difficult. Thus, type-I method employs a priori preventive adjustments before the buffer gets full using $\Delta B(t)$ to reduce the occurrences of the buffer fullness. **Fig. 11** shows the comparison of the rate control results over time between type-I and type-II method. After 5.6 seconds, the amplitude of sending rate adjustments by type-II method is too big and it fluctuates too much. This was because the system did not escape intelligently from the overflow state. If the sending rate is too high, the possibility of losing packets also becomes high. However, our proposed approach, type-I method, showed stable performance over the entire simulation period in that it generated moderate amplitude of sending rate adjustments and did not enter overflow states. In **Fig. 11**, the dotted curve represents the results by no-control (without rate control) which is TCP Reno. For following experimentations, the type-I method has been used for rate control method.

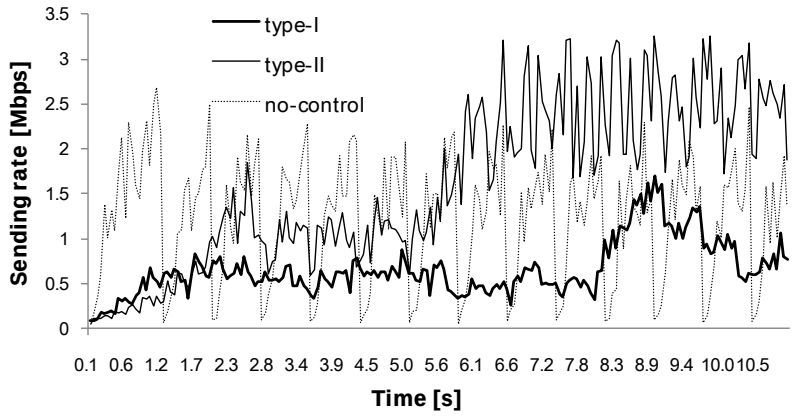

Fig. 11. The comparison result of the sending rate for type-I and type-II method (with the buffer size of 32,768 bytes).

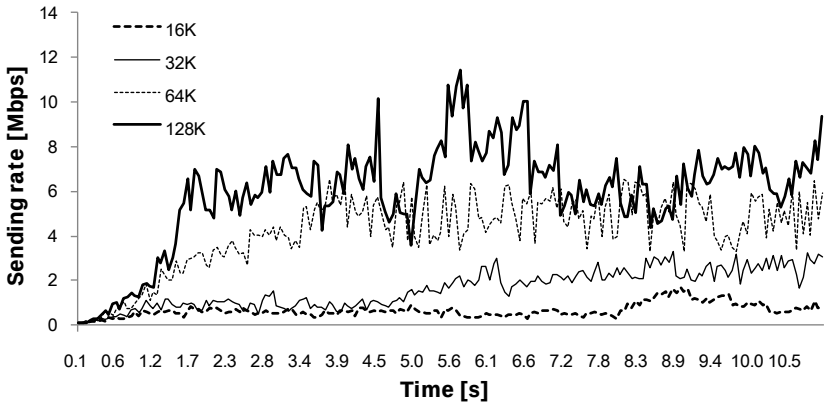

Fig. 12. The sending rate with different buffer size.

As shown in **Fig. 12**, the sending rate control varies according to the different buffer size. As the buffer size gets to be smaller, rate control becomes smoother and thus the reaction becomes slower to the bandwidth changes. Then, this will result in under-utilization of the wireless channel.

The **Fig. 13**, **Fig. 14**, and **Fig. 15** show the simulation results how the buffer awareness of the proposed approach protects the client's buffer from overflow and underflow. To see the impact of buffer underflow, the packet arrival rates are compared between the proposed approach and the TFRC. In this simulation, we set the packet consumption rate of the client to 8Kb/s and the client's buffer size to 16Kbytes. The maximum sending rate was 64Kb/s and the packet size was 160 bytes. In addition, to purposely cause buffer underflow, we set up heavy network traffic; 48 TCP connections are competing for their own separate media streaming flows. This means the average number of packets each client can receive every second is at most 8. As you can see in **Fig. 13**, the packet arrival rate of our proposed approach is faster than that of TFRC by 2.4 times in average. Because there were no moments that packet arrival rate was zero, the client buffer did not experience buffer underflow (buffer emptiness). Also, we measure the fraction of packet loss caused by the buffer overflow out of the entire losses experienced by the media flow. The overall loss rate increases with the number of competing TCP flows as shown in **Fig. 14**. **Fig. 15** shows the percentage of the loss rate caused by buffer

fullness out of the overall loss rate of **Fig. 14**. It also shows that the percentage of dropped packets due to buffer fullness decreases with the number of competing TFRC and type-I flow. This is because the throughput that the bottleneck link allows decreases and hence less and less received packets need to be buffered. The **Fig. 15** clearly shows that the TFRC consistently results in buffer overflows and hence it increases the number of dropped packets due to the buffer fullness. Please note that the network traffic becomes heavy as the number of TCP flow increases, and thus the occurrence of buffer fullness, i.e. the packet loss, becomes smaller.

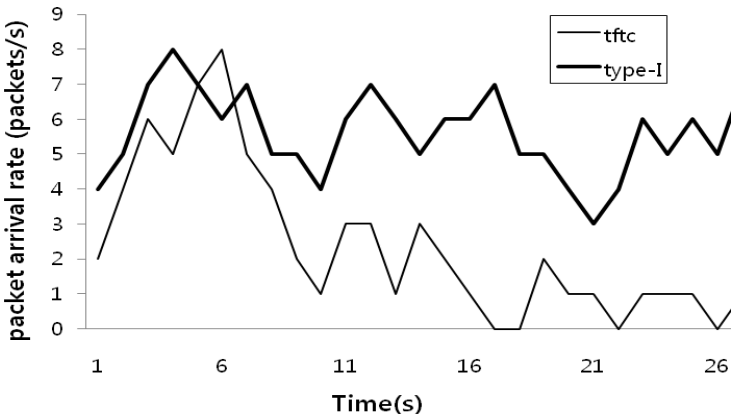

Fig. 13. The packet arrival rate at the receiver.

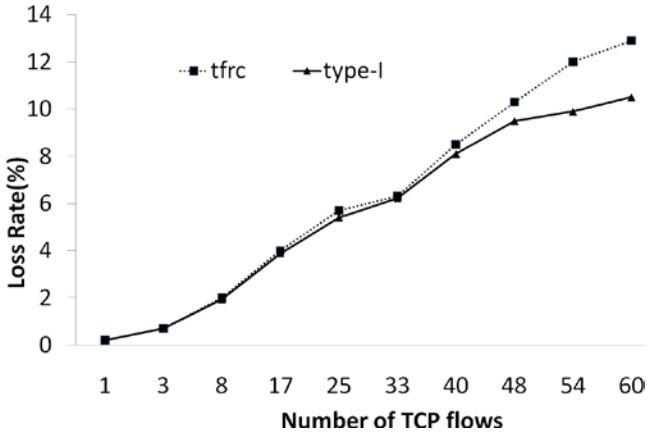

Fig. 14. Overall loss rate experienced by TFRC and type-I flows

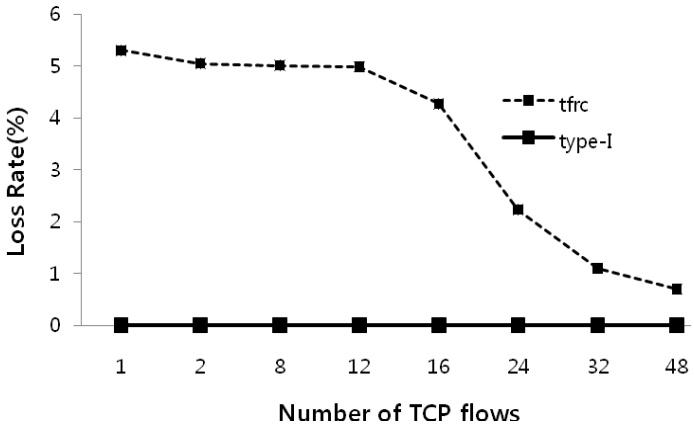

Fig. 15. Loss rate due to buffer overflow

To validate our proposed approach in a real system, we built a prototype system on PDAs with the proposed adaptive rate control algorithm. For this test, the resolution of a PDA for displaying video was set to 240×160 as shown in **Fig. 16**. A MPEG-2 encoder is used for video coding. The pre-defined frame rate is set to 20 frames per second (fps) at the server, but the achieved frame rate of the PDA with the socket buffer size of 16,384 bytes was about 16.2-16.5 fps. The expected bit rate in this experiment was 64Kbps. Note that for simplicity and without loss of generality, single PDA was used as a receiver. When the packet size was 1,500 bytes, obtained bandwidth and round trip time were 426,300 bps and 19ms, respectively. The overall shape of the measured discrete buffer level follows the curve of simulation results shown in **Fig. 9**. But, the shape of the rate variance was a little different from **Fig. 9**. The reason was that the time required to adjust the rate in the PDA affected the latency of rate adjustment and responsiveness. However, the proposed method worked well as expected without buffer overflow at the PDA device.

Fig. 16. The snapshots of the playback of streaming video on PDAs

5. Conclusions

In this paper, we proposed an adaptive rate control scheme for video streaming over wireless channel, which is controlled by the status of the buffer at clients. Since client devices (receiver) have limited computing capabilities and limited amount of buffer capacities, the socket buffer in receiver can be frequently overflowed even if there were no network congestions or BER of wireless channel was low. We exploited the fact that the rate at the server is strongly affected by the receiver's state. We proposed a simple prediction for controlling the sending rate at the server by inspecting the current buffer level at the receiver side. In addition, a simple cost effective rate control was presented and it works just by varying the inter-packet delays at the server, without requiring any modifications of network protocols.

In the evaluation of the proposed scheme in simulations and a prototype system as well, we showed the occurrences of the overflowed socket buffer could be kept very low. Therefore, the proposed method can maintain smoother rate control and react to bandwidth changes promptly.

In addition to rate control of wireless video streaming, it is also important bit rate control for quality and performance of video, and it has important implications in wireless video streaming field. The research on the video quality at the received video is part of future work and it should be thoroughly investigated for video performance (e.g. PSNR) with different rate controls including bit-rate and video coding methods.

References

[1] Sally Floyd, Mark Handley, Jitendra Padhye and Jörg Widmer, "Equation-based congestion control for unicast applications," in *Proc. of ACM SIGCOMM Computer Communication Review*, pp. 43-56, 2000. Article (CrossRef Link)

[2] Jitendra Padhye, Victor Firoiu, Don Towsley and Jim Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proc. of ACM SIGCOMM Computer Communication Review*, pp. 303-314, 1998. Article (CrossRef Link)

[3] Minghua Chen and Avideh Zakhor, "Rate control for streaming video over wireless," *IEEE Wireless Communications*, vol. 12, no. 4, pp. 32-41, 2005. Article (CrossRef Link)

[4] Dorgham Sisalem and Henning Schulzrinne, "The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme," in *Proc. of Int. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 1998. Article (CrossRef Link)

[5] Song Cen, Pamela C. Cosman, and Geofrrey M. Voelker, "End-to-end differentiation of congestion and wireless losses," *IEEE/ACM Transaction On Networking*, vol. 11, no. 5, pp. 703-717, 2003. Article (CrossRef Link)

[6] Byunghun Song, Kwangsue Chung and Yongtae Shin, "SRTP: TCP-friendly congestion control for multimedia streaming," in *Proc. of 16th Int. Conference on Information Networking*, pp. 529-538, 2002. Article (CrossRef Link)

[7] Bing Zheng and Mohammed Atiquzzaman, "A novel scheme for streaming multimedia to personal wireless handheld devices," *IEEE Transaction on Consumer Electronics*, vol. 49, no. 1, pp. 32-40, 2003. Article (CrossRef Link)

[8] Guang Yang, Mario Gerla and M. Y. Sanadidi, "Adaptive video streaming in presence of wireless errors," in *Proc. of 7th IFIP/IEEE International Conference, MMNS*, pp. 26-38, 2004. Article (CrossRef Link)

[9] Marwan M. Krunz and Mohamed Hassan, "Adaptive rate control scheme for video streaming over wireless channels," in *Proc. of the Data Compression Conference*, pp. 242-251, 2004. Article (CrossRef Link)

[10] Giovanni Gualdi, Rita Cucchiara and Andrea Prati, "Low-latency live video streaming over low-capacity networks," in *Proc. of the Eighth International Symposium on Multimedia (ISM '06)*, pp. 449-456, 2006. Article (CrossRef Link)

[11] Panagiotis Papadimitriou, Vassilis Tsaoussidis and Chi Zhang, "Enhancing video streaming delivery over wired/wireless networks," in *Proc. of European Wireless Conference*, 2007. Article (CrossRef Link)

[12] Bing Zheng and Mohammed Atiquzzaman, "Multimedia Over High Speed Networks: Reducing Network Requirements with Fast Buffer Fillup," in *Proc. of Global Telecommunications Conference, GLOBECOM 98*, pp. 779-784, 1998. Article (CrossRef Link)

[13] Lachlan L. H. Andrew, Tony Cui, Jinsheng Sun, Moshe Zukerman, King-Tim Ko and Sammy Chan, "Buffer sizing for nonhomogeneous TCP sources," *IEEE Communication Letters*, vol. 9, no. 6, pp. 567-569, 2005. Article (CrossRef Link)

[14] The network simulator ns-2, Information Sciences Institute, The University of Southern California, 2006.
[15] Youn-Sik Hong, Hwa-Seok Lim, Cheol-Ho Lee and Seung-Sik Choi, "A receiver based rate control scheme for streaming video over wireless," in *Proc. of IEEE Systems, Man and Cybernetics (SMC)*, pp. 2297-2302, 2009. Article (CrossRef Link)
[16] Ji-Hong Kim, Yong-Hyun Kim, Youn-Sik Hong and Ki-Young Lee, "Performance Analysis of TCP Downstream Between Heterogeneous Terminals in an Infrastructure Network," in *Proc. of Computational Science and Its Applications (ICCSA)*, LNCS 4706, Part II, pp. 778-789, 2007. Article (CrossRef Link)

Youn-Sik Hong is currently a Professor of Computer Science and Engineering, University of Incheon. He received Ph.D degree from Korea Advanced Institute of Science and Technology (KAIST) in 1989. His current research interests include wireless communication and mobile computing.

Heemin Park is an assistant professor in the department of multimedia science at Sookmyung Women's University, Seoul, Korea. He received his B.S. and M.S. degrees in computer science from Sogang University, Korea in 1993 and 1995, respectively, and the Ph.D. degree in electrical engineering from the University of California, Los Angeles, in 2006. Before joining Sookmyung Women's University, he worked at Samsung Electronics in Korea as a principal engineer and technical leader of the image processor design group for mobile phone cameras. His research interests include a broad range of multimedia, ubiquitous & entertainment computing, and cloud computing with focus on imaging and sensing technologies.