Efficient Packetization for DCCP Flows over ARQ-based Wireless Networks

Chi-Huang Shih[1] Chih-Heng Ke[2, *] Yeong-Yuh Xu[1]

[1] Department of Information Engineering and Computer Science, Hung Kuang University
Taichung 407, Taiwan, ROC
{chshih,yyxu}@sunrise.hk.edu.tw

[2] Department of Information Engineering and Computer Science, National Quemoy University
Quemoy 892, Taiwan, ROC
smallko@gmail.com

Received 5 February 2013; Revised 31 March 2013; Accepted 9 April 2013

Abstract. Datagram congestion control protocol (DCCP) is a transport-layer protocol designing for multimedia applications to transmit congestion-controlled data streams over Internet. The major features of DCCP protocol include a TCP-friendly rate control (TFRC) mechanism to ensure fair bandwidth sharing with TCP, and a partial payload protection mechanism to retrieve the effective data from error-corrupted packets. In enabling a robust data transmission for DCCP flows over wireless networks with the automatic repeat request (ARQ) capacity, the packetization scheme needs to be fitted to the new DCCP features and ARQ-based transmission scenario. To be more specific, the partial payload protection mechanism can be inefficient owing to MAC-layer retransmissions, while varying packet size has an adverse effect on the TCP-friendliness of DCCP flows. This paper proposes two main contributions in designing the efficient packetization scheme for DCCP flows, namely (1) a DCCP protocol enhancement to benefit the partial payload protection mechanism by utilizing the ARQ retransmissions and then (2) an in-packet segmentation scheme to improve the data goodput while preserving the TCP-friendliness of the DCCP flows. The experimental results demonstrate the effectiveness of the proposed scheme based on the results of packet corruption probability and normalized throughput under a variety of network conditions.

Keywords: DCCP, packetization, ARQ, wireless Internet

1 Introduction

In general, multimedia streaming applications require the support of stringent bandwidth, delay and loss constraints in order to guarantee a satisfactory perceptual quality of the media content at the receiver end. While these constraints can be satisfied relatively easily over wired connections, an increasing number of users are now choosing to access the Internet via wireless devices such as lap top computers, PDAs, cell phones, and so forth. Wireless channels are inherently lossy, and thus in executing multimedia streaming applications over the wireless Internet, the transmission system must be capable of dealing not only with congestion-related losses and delays, but also with the wireless error packet losses.

The Internet Engineering Task Force (IETF) created a new transport-layer protocol designated as the datagram congestion control protocol (DCCP) for the congestion-controlled transportation of datagrams in delay-sensitive applications [1]. Compared with the family members such as transmission control protocol (TCP) and user datagram protocol (UDP) in the network protocol stack, DCCP supports the establishment of reliable connections and two unique features, namely TCP-friendly congestion control and partial payload protection. TCP-friendly congestion control aims at adjusting the data rate of a media flow similar to that of a typical TCP flow along the same path under the same network conditions such that TCP-friendly flows experience minimal packet losses and end-to-end delays among best-effort Internet traffic [2]. Specifically, DCCP is compliant with various TCP-friendly congestion control mechanisms, including TCP-like congestion control [3] and TCP-friendly rate control (TFRC) [4], [5]. Compared with the TCP-like congestion control mechanism, TFRC has a smoother transmission rate change under the various network situations [6], [7]. According to the results of performance analysis presented in [8], throughput fairness between TFRC and TCP flows can be preserved under various traffic load conditions. However, in the heavy traffic load condition, TFRC flows show a greater variance between their throughputs than TCP flows do. Based on the transport rate constraint, the multimedia applications must be capable of quickly adapting the output data rate to changes in the DCCP transmission [9], [10]. This implies that applications with media content adaptation best fit DCCP to fully exploit the advantages of DCCP

[*] Corresponding author.

transport in terms of low delay and packet loss rate. For applications without the capability of adaptive media coding, however, UDP still suits such applications since DCCP might defer the packet delivery during periods of network congestion and the multimedia streaming quality can be degraded accordingly due to the late data arrival.

On the other hand, the perceived quality of the reconstructed media content at the receiver end is highly sensitive to data losses, and thus both the media coding scheme and the transport-layer protocol must utilize some form of error handling mechanism to enhance the transmission robustness. Contemporary video coding standards such as H.263+, H.264 and MPEG-4 recommend an explicit set of error resiliency tools to ensure a high loss resiliency over noisy transmission channels. For example, MPEG-4 utilizes a reversible variable length coding (RVLC) scheme to retrieve useful information from any corrupted data recovered at the decoder end [11]. To support the error resiliency mechanisms, DCCP enables a partial payload protection capability which is similar to the UDP-Lite protocol [12], to forward any packets corrupted by transmission errors to the upper receiving application.

Although DCCP ensures that the corrupted packets are at least processed by the streaming application rather than being simply dropped, in the case of wireless error bursts, the decoder is still unable to recover any meaningful information from the data and thus a noticeable quality degradation occurs [13]. To enhance the transmission robustness, the packetization schemes mitigate data impairment during transmission to increase the data goodput by controlling the transport packet sizes. In [14], it was shown that a good balance could be achieved between the transmission efficiency and the robustness of the received media content by utilizing a packetization algorithm in which the packetization threshold was determined in accordance with the bit error rate and the currently available bandwidth. Reviewing the literature, it is found that several DCCP-based schemes have been proposed for the transmission of H.264 or MPEG-4 video streams over the Internet. In general, these schemes adopt adaptive video coding [10], [15] or error control mechanisms [16], [17] at the application layer to ensure the quality of the reconstructed video. With the features of TFRC and partial payload protection, however, new challenges occur when implementing DCCP-based packetization schemes over wireless environments. To ensure the robust transmission quality, most wireless networks utilize the automatic repeat request (ARQ) mechanism in MAC layer to retransmit the error-corrupted packets with a pre-defined transmission times. While a corrupted packet is forwarded by DCCP to the upper application, the current DCCP protocol discards its following retransmissions since the multiple packets with the same sequence number are typically regarded as duplicates. This leads to the inefficient bandwidth consumption even through the last retransmission can be completely received without any errors.

Additionally, the packet size control in the traditional packetization schemes easily causes a throughput bias problem for DCCP flows. The use of a large packet size improves the bandwidth efficiency, but increases the data corruption rate under poor channel conditions. Conversely, using a small packet size reduces the degree of packet corruption. However, the conventional TFRC protocol is intended only for streaming applications with a fixed packet size, i.e. it achieves a congestion control function by varying the rate at which the individual packets are transmitted rather than by varying their size. Since the fair bandwidth sharing depends on the packet size, DCCP/TFRC flows which use a small packet size to transmit their data only achieve a fraction of the throughput achieved by flows using a larger packet size. In [18], the small-packet variant of TFRC (TFRC-SP) was proposed for small-packet flows to fairly compete for the network bandwidth with larger-packet flows. More specifically, TFRC-SP was intended for flows that need to send frequent small packets, and enforced a minimal interval of 10 milliseconds between packets. The typical examples for TFRC-SP were audio/voice applications such as Voice over IP (VoIP). For streaming flows with bulk data, Widmer et al. in [19] suggested a packetization strategy in which the TFRC throughput was computed on the basis of a large packet size, but the data were actually transmitted over the network in small packets. However, this strategy causes a higher packet rate combined with the under-estimated loss rate and therefore, a strong throughput bias in favor of sending small packets at a high rate can be observed. To cope with this bias problem, rate correction schemes are required to compensate for the effects of throughput bias for TFRC flows with different packet sizes. Therefore, despite the contributions of the studies presented above, in implementing DCCP-based transmission system, the challenge still remains to maximize the robustness and efficiency of the packetization scheme while simultaneously maintaining the throughput fairness.

In an attempt to resolve the challenge described above, this paper considers an enhancement of the existent DCCP protocol to deal with the MAC-level retransmissions, and an in-packet segmentation scheme combined with the wireless ARQ protocol is proposed to improve the data goodput over wireless channels while simultaneously maintaining their TCP-friendliness. In the proposed segmentation scheme, the payload of each transport packet is partitioned into virtual segments with a size determined by the channel status in such a way as to maximize the transmission efficiency. By means of a partial payload protection mechanism, the in-packet segmentation scheme detects any corrupted segments at the receiver end and then replaces these segments with the corresponding segments within the subsequently-received MAC-level retransmissions. The in-packet segmentation scheme has two principal advantages, namely (1) it results in virtually no change in the packet size, and therefore preserves the TCP-friendly nature of the video stream; and (2) the segment recovery technique benefits the error-

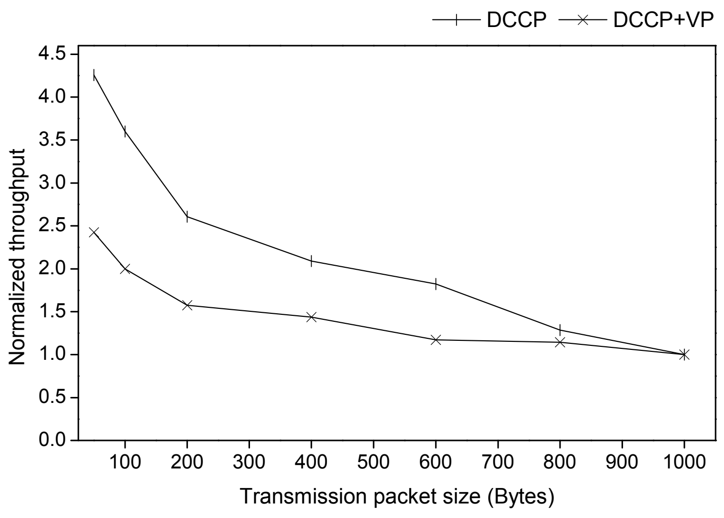


Fig. 1. Fairness with various transmission packet sizes

tolerant applications since more useful data within the corrupted packet and even a complete packet can be obtained after wireless retransmissions.

The remainder of this paper is organized as follows. Section 2 describes the TFRC operation and the throughput bias problem associated with packet sizing. Section 3 introduces the DCCP protocol enhancement in ARQ-based wireless networks and Section 4 presents the proposed in-packet segmentation scheme. After evaluating the proposed scheme based on the experimental results in Section 5, the paper concludes in Section 6.

2 TFRC and Throughput Bias Problem

TFRC is a rate-based protocol designed for unicast flows that co-exist with TCP traffic over the Internet. The TFRC receiver is responsible for reporting on feedback involving the instantaneous network dynamics to the sender at least once per round trip time (RTT). The sender then adjusts the transmission rate accordingly. TFRC uses a throughput equation to estimate the maximum permissible sending rate, T, as a function of the loss event rate and the RTT:

$$T = \frac{S}{r\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \quad , \qquad (1)$$

where:
T is the transmit rate in bytes/sec;
S is the packet size in bytes;
r is the round-trip time in seconds;
p is the current loss event rate (between 0 and 1.0) of the number of loss events as a fraction of the number of packets transmitted, and
t_{RTO} is the TCP retransmit timeout value in seconds.
In TFRC, a loss event contains one or more losses occurred during one RTT, and several losses appearing in the same RTT are therefore treated as a single loss event. Because the loss bursts are grouped in the same loss event, TFRC can avoid abrupt oscillations in the transmission rate while being responsive to congestion.

Significantly, the TFRC standard, RFC3448, only targets applications with fixed packet size, and TFRC performs the congestion control function by means of varying the sending rate in packets per second. To understand the relation between the DCCP transmission packet size and the throughput bias for the integrity of our presentation, we implement the ideas presented in [19] and reproduce the experiment results. The experiment set-up is described in Section 5. In the experiment, DCCP utilizes the packet size of 1000 bytes to calculate the TFRC sending rate and the actual packet size used for transmission is varied from 50 to 1000 bytes. Additionally, the "virtual packets (VP)" rate correction mechanism proposed in [19] is applied to the DCCP to observe its improvement in removing the throughput bias. Figure 1 shows the experimental results for DCCP with and without the VP mechanism. The throughput of the schemes is normalized to the throughput of a DCCP flow using the

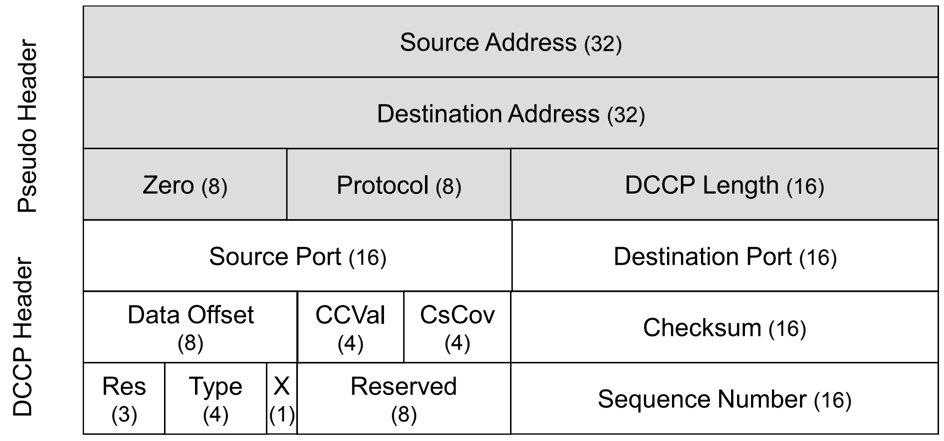


Fig. 2. DCCP packet format. All data are aligned on a 32-bit boundary and the number inside brackets represents the field length in bits

packet size of 1000 bytes in both delivering packets and calculating the TFRC rate. The higher normalized throughput indicates a severe throughput bias. In Figure 1, we plot the normalized throughput as a function of the transmission packet size. As the transmission packet size is decreased, the DCCP without the “virtual packets” mechanism becomes more aggressive to achieve a higher throughput than that of the DCCP using the packet size of 1000 bytes and is thus further from fair. The DCCP with the “virtual packets” mechanism has the similar observations as the DCCP without the “virtual packets” mechanism but improves TFRC throughput fairness. From the experimental results presented in Figure 1, it can be clearly observed that when the transmission packet size decreases, it becomes more difficult for the rate correction mechanism to mitigate the throughput bias effects. In other words, as the transmission packet size approaches to the original one adopted for TFRC rate calculation (in this case, 1000 bytes), the TFRC fairness can be better preserved. Such an observation is important in this study to develop our fair and robust packetization scheme for DCCP-based data transmissions.

3 DCCP Protocol Enhancement for MAC-layer ARQ

Figure 2 illustrates the DCCP packet header. The checksum coverage field (CsCov) specifies the parts of the packet which are covered by the checksum field. For example, full coverage includes the DCCP header, the network-layer pseudo header and the payload (i.e., the application data), whereas minimum coverage includes only the DCCP header and the network-layer pseudo header. Based on the partial checksum coverage, the DCCP packet is divided into the insensitive part and the sensitive part which is covered by the checksum. The sensitive part typically includes vital information such as headers and application-specific identification. Consequently, packets with errors in the sensitive part are discarded since either the network protocol stack or the target application itself cannot appropriately process the corrupted packets. On the other hand, DCCP and the lower link layer ignore any errors within the insensitive part of the packet, and the error-tolerant application (e.g., voice and video) can utilize the damaged DCCP packets to improve the media presentation quality by means of retrieving useful information from the corrupted data. It is noted that the IP layer has no checksum to cover the IP payload and therefore delivers any packets with correct IP header to the upper layers.

Since wireless MAC layers generally employ an ARQ mechanism to retransmit damaged packets, the DCCP implementation has to deal with the packet forwarding for a series of retransmissions. In order to avoid the network attack and duplicates, a lower bound to the amount of the similar packets is required and its value should not exceed the MAC-layer transmission times. After the pre-defined lower bound is attained, DCCP ignores any newly-incoming packets with the same sequence number. In the popular 802.11 wireless local area network, the retransmission limit used for long packet length and short packet length are 4 and 7, respectively, and the corresponding maximum transmission times are thus 5 and 8. Since DCCP fits the video applications, which usually prefer using the large packets to obtain a fair bandwidth share among TCP connections, the value of the lower bound is currently set to 5 in this paper.

4 In-packet Segmentation Scheme with Wireless ARQ Protocol

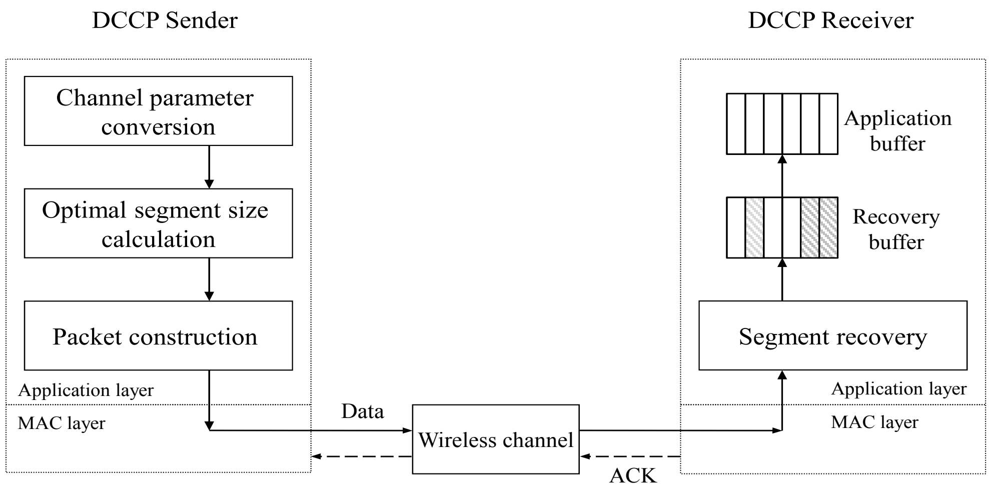


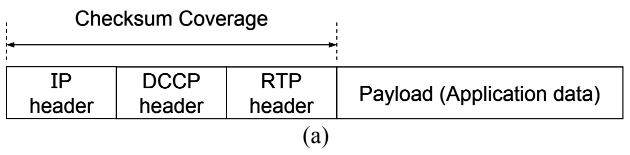
Fig. 3. System diagram of the in-packet segmentation scheme

In the DCCP transmission system, an in-packet segmentation scheme is developed to improve the data good-put in the application domain while simultaneously maintaining the TFRC throughput fairness in the transport domain. It is noted that in order to appropriately packetize the application data, the in-packet segmentation scheme operates at the application layer and in an end-to-end manner. As shown in Figure 3, the application packet is partitioned into segments at the sender end, and a segment recovery process combined with the wireless ARQ protocol is performed at the receiver end to replace any corrupted segments in the original packet with the corresponding segments in the retransmitted packet(s). Specially, a recovery buffer is utilized to record the seg-ment status of partially-corrected packets for segment replacement purpose, while an application buffer stores the error-free segments. In segmenting the application packet, the proposed scheme first obtains the bit-level channel information through a parameter conversion procedure. Given the current wireless channel conditions, the segment size which achieves the optimal balance between the error robustness of the transmitted data and the transmission efficiency can be determined. Following the segment recovery process, the application receives either a complete packet or a corrupted packet containing fewer wireless errors than the original corrupted packet. Importantly, the proposed partitioning scheme results in no more than a minor change in the original packet size, and thus the throughput bias problem induced under the TFRC protocol by packets with a variable size is avoid-ed. Consequently, the TCP-friendliness of the video flows is preserved. The details of the proposed scheme are presented in the paragraphs below.

4.1 Packet Structure

Figures 4(a) and 4(b) show the original DCCP packet format and the DCCP packet format under the proposed in-packet segmentation scheme, respectively. In both cases, the DCCP packet contains two parts, i.e. a sensitive part protected by the checksum coverage mechanism and an insensitive, unprotected part. Real-time protocol (RTP) is located at the application layer to facilitate the delivery of multimedia flows in terms of stream syn-chronization and transmission statistics monitoring [20]. As shown in Figure 4(b), the virtual segmentation (VS) header includes a one-byte "Ratio" field to indicate the number of segments within the packet, and multiple two-byte "Checksum" fields (one field per segment) to identify the corruption status of the corresponding segments. As shown in the upper schematic in Figure 4(b), the VS header is added immediately behind the sensitive part of the packet. Moreover, it can be seen that the original checksum coverage is extended such that it also covers the "Ratio" field of the VS header.

At the receiver end, any packet having at least one corrupted segment (as indicated in the VS header) is re-garded as a wireless loss. Meanwhile, congestion-induced packet losses are detected by examining the sequence number in the RTP header of the incoming packets. The ability to differentiate between wireless losses and con-gestion losses can help the media application itself to react to the different loss types by appropriately adjusting the parameters of source coding and/or channel coding. Although any packets which contain wireless errors within the sensitive part are dropped in DCCP layer and are therefore misclassified as congestion losses, the proposed virtual segmentation technique nevertheless provides an efficient error detection mechanism for the support of loss differentiation in wireless Internet channels.



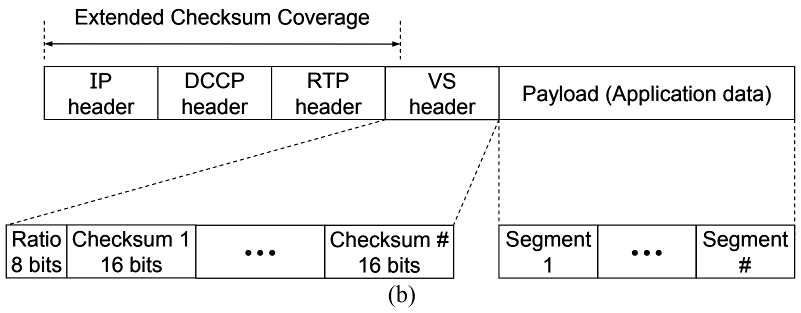


Fig. 4. DCCP packet formats with and without virtual segmentation: (a) Original DCCP packet (i.e. non-segmented); (b) DCCP packet with virtual segmentation

4.2 Virtual Segment Size for Bursty Channels

This subsection describes the mechanism used in the proposed in-packet segmentation scheme to calculate the segment size which optimizes the transmission efficiency for a given channel bit error rate and burst bit error length. In calculating the segment size, the burst error pattern over the transmission channel is modeled using a Gilbert model (or a two-state Markov model), as shown in Figure 5. The Gilbert model has just two states, i.e. a Good-state and a Bad-state. The bit errors are generated in accordance with an average bit error rate of P_b and an average burst bit error length of L_b. In the Good-state, a bit is damaged with a probability of 0, while in the Bad-state, a bit is damaged with a probability of 1. The transition probabilities P_{gb} and P_{bg} are derived in accordance with the values of P_b and L_b as follows:

$$P_{bg} = \frac{1}{L_b}, \quad P_{gb} = P_{bg} \times \frac{P_b}{1-P_b} . \tag{2}$$

Let L_{seg} be the segment length in bits and let L_{vs} be the VS header length in bits. Based on the bit-level Gilbert channel model, the segment error rate P_B can be calculated as:

$$\begin{aligned} P_B &= 1 - \left((1-P_b) \times (1-P_{gb})^{L_{seg}+L_{vs}-1}\right) \\ &= 1 - \left((1-P_b) \times \left(1 - \frac{P_b}{L_b(1-P_b)}\right)^{L_{seg}+L_{vs}-1}\right) . \end{aligned} \tag{3}$$

The transmission efficiency for the segment is therefore given by

$$\begin{aligned} \mathrm{E} &= \left(\frac{L_{seg}}{L_{seg}+L_{vs}}\right) \times (1-P_B) \\ &= \left(\frac{L_{seg}}{L_{seg}+L_{vs}}\right) \times \left[(1-P_b) \times \left(1 - \frac{P_b}{L_b(1-P_b)}\right)^{L_{seg}+L_{vs}-1}\right] . \end{aligned} \tag{4}$$

Given the channel bit error rate and the burst bit error length, and assuming L_{seg} to be a continuous variable, the optimal segment size required to maximize the transmission efficiency can be obtained by differentiating Eq. (4) with respect to L_{seg} and setting the derivative equal to 0, i.e.

$$\frac{d}{dL_{seg}} E(L_{seg}) = \frac{d}{dL_{seg}} \left[\left(\frac{L_{seg}}{L_{seg}+L_{vs}}\right) \times \left[(1-P_b) \times \left(1 - \frac{P_b}{L_b(1-P_b)}\right)^{L_{seg}+L_{vs}-1}\right]\right] = 0 . \tag{5}$$

Accordingly, the optimal segment size L_{opt} is given by

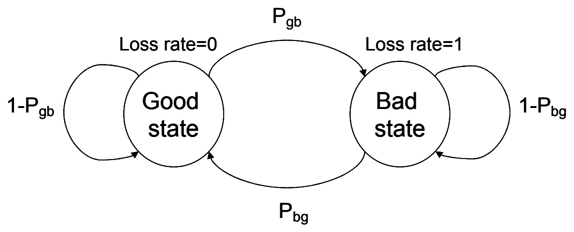


Fig. 5. Gilbert model

$$L_{opt} = \frac{-L_{vs}+\sqrt{L_{vs}^2-\frac{4L_{vs}}{w}}}{2}, \quad \mathrm{w} = \ln\left(1-\frac{P_b}{L_b(1-P_b)}\right). \tag{6}$$

Figure 6 (a) shows the optimal segment size calculated using Eq. (6) with L_b=20 and P_b=10^{-1}~10^{-6}. As the bit error rate is increased, the resulting optimal segment size is decreased to ensure the robust transmission. Figure 6 (b) shows the optimal segment size with P_b=10^{-3} and L_b=20~200. As the burst bit error length increases, the more errors easily aggregate within the packet and the segment error rate is accordingly decreased. From Figure 6 (b), the longer burst bit error length leads the larger optimal segment size to achieve higher transmission efficiency.

As shown in Eq. (6), to determine the optimal segment length, it is first necessary to establish the values of P_b and L_b. Since both the segment error rate $\mathrm{P_B}$ and the burst segment error length $\mathrm{L_B}$ can be easily estimated at the application layer, a simple conversion method suffices to derive P_b as a function of P_B and L_B. Based on the Gilbert channel model, L_B can be expressed as

$$L_B = \frac{P_B}{\left[1-\left(1-\frac{P_b}{L_b(1-P_b)}\right)^{Lseg+Lvs}\right]\times(1-P_B)} . \tag{7}$$

Combining Eq. (7) with Eq. (3), P_b is given by

$$P_b = 1-\frac{1-P_B}{\left(1-\frac{L_B}{L_B(1-P_B)}\right)^{\frac{Lseg+Lvs-1}{Lseg+Lvs}}} . \tag{8}$$

Similarly, L_b can be calculated as

$$L_b = \frac{P_b}{(1-P_b)\left[1-\left(1-\frac{L_B}{L_B(1-P_B)}\right)^{\frac{1}{Lseg+Lvs}}\right]} . \tag{9}$$

To derive L_b as a function of P_B and L_B, substituting Eq. (8) into Eq. (9) yields

$$L_b = \frac{Q^{Lseg+Lvs-1}-(1-P_B)}{(1-P_B)(1-Q)}, \quad Q = (1-\frac{P_B}{L_B(1-P_B)})^{\frac{1}{Lseg+Lvs}} . \tag{10}$$

Based on the conversion from segment-level parameters (P_B, L_B) to bit-level parameters (P_b, L_b), the virtual segment size can be calculated by using Eq. (6) to achieve the high transmission efficiency over bursty channels.

4.3 Segment Recovery Mechanism

In the in-packet segmentation scheme, the partial checksum coverage feature and the MAC ARQ protocol are utilized to accomplish a segment recovery mechanism to maximize the application goodput of DCCP flows. The proposed approach uses two data buffers for DCCP-received packets, namely the recovery buffer and the application buffer. In the recovery buffer, the corrupted segments of partially-corrected packets are replaced with the corresponding segments within the duplicate packets received in the subsequent ARQ-requested MAC-layer retransmissions. After the segment recovery, the processed packet is forwarded to the application buffer for future processing specified by the application itself, and the follow-up copies of the same packet will be discard

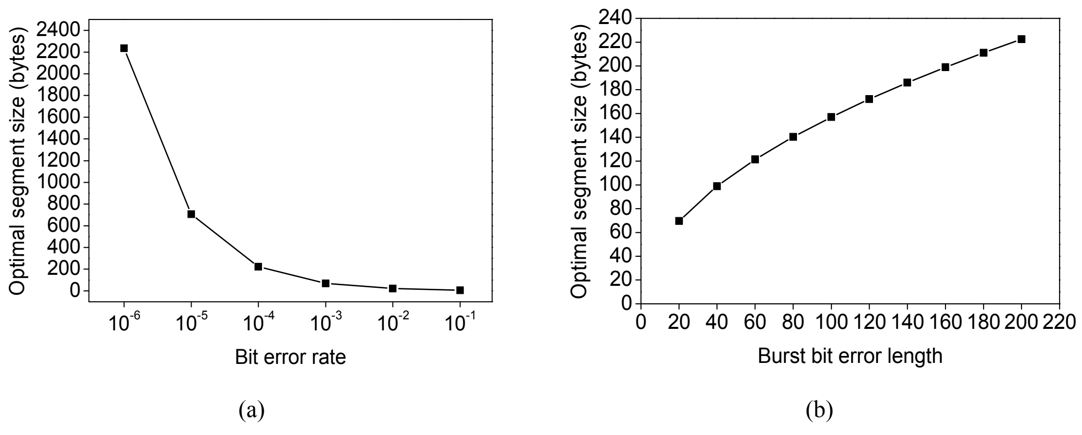


Fig. 6. Optimal segment size for varied bit error rate (a) and varied burst bit error length (b). In (a), the burst bit error length is set to 20 bits and the bit error rate is 10^{-3} in (b)

ed. This immediate packet forwarding aims at facilitating the timely presentation of application content. Let the number of MAC frame transmissions be denoted as M_{cur} and let the initial value of M_{cur} when transmitting a MAC frame be set to 0. The major steps in the formal wireless ARQ procedure can be summarized as follows:

1) The sender transmits a frame to the receiver and the frame transmission counter M_{cur} is incremented by 1 (i.e., $M_{\mathrm{cur}}=M_{\mathrm{cur}}+1$).
2) The receiver examines the correctness of the received frame. If the frame is correctly received, an ACK is sent back to the sender.
3) The receiver forwards the received frame to the upper layer.
4) If the ACK is not received and M_{cur} is less than the MAC-layer transmission limit M, the process returns to Step 1. Otherwise, the process terminates.

Based on the MAC-level ARQ protocol, the DCCP packets received at the receiver end are processed using the segment recovery algorithm in accordance with the following steps:

1) The receiver receives a packet with N_{seg} virtual segments from the DCCP/MAC layer.
2) If the received packet is a new packet and a previously-received packet is already stored in the recovery buffer, the stored packet is forwarded to the application buffer.
3) If the received packet is not a new packet and no stored packet is found in the recovery buffer, the received packet is discarded and the process returns to Step 1.
4) The receiver examines the correctness of the segments within the received packet. If all the segments are correctly received, the packet is passed to the application buffer and the process returns to Step 1. Otherwise, the receiver saves a copy of the corrupted packet to the recovery buffer and performs a number of different actions depending on the current packet transmission status:
   (a) If the currently received packet is a new packet, the receiver records the positional information relating to the corrupted segments and sets the current number of corrupted segments equal to N_{err}.
   (b) If the currently received packet is a retransmission of a previously-transmitted corrupted packet, the receiver replaces each corrupted segment in the original corrupted packet with the corresponding segment in the currently received segment (if it is correct), and decrements N_{err} by a value of one, i.e. $N_{\mathrm{err}}=N_{\mathrm{err}}-1$, each time it makes a replacement.
5) When N_{err} reduces to zero, i.e. $N_{\mathrm{err}}=0$, the packet is forwarded to the application buffer. The process returns to Step 1.

It is noted that in Step 3, the segment recovery mechanism needs to remove the additional packet retransmissions after the complete application packet is obtained. To further optimize the recovery process, the cross-layer architecture can be used to enable the information exchange among application, DCCP and MAC layers such that adjusting the retransmission times is possible [21].

5 Performance Evaluation

As shown in Figure 7, the experimental environment contained six hosts, one Cisco 2600 router, and one network bridge. The wired links between each node were connected via Fast Ethernet and the bandwidth of the bottleneck link was specified as 1 Mbps. All the links were Drop-Tail links and the queue length at the bottleneck link was set to four times the bandwidth-delay product. The network bridge incurred a transfer delay of 25

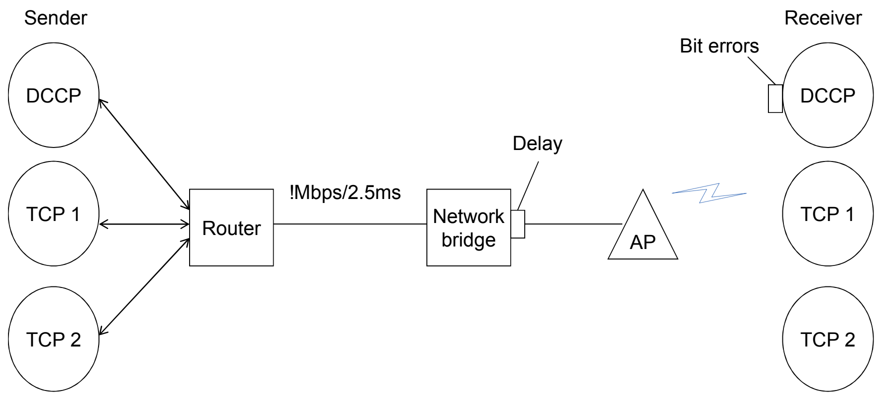


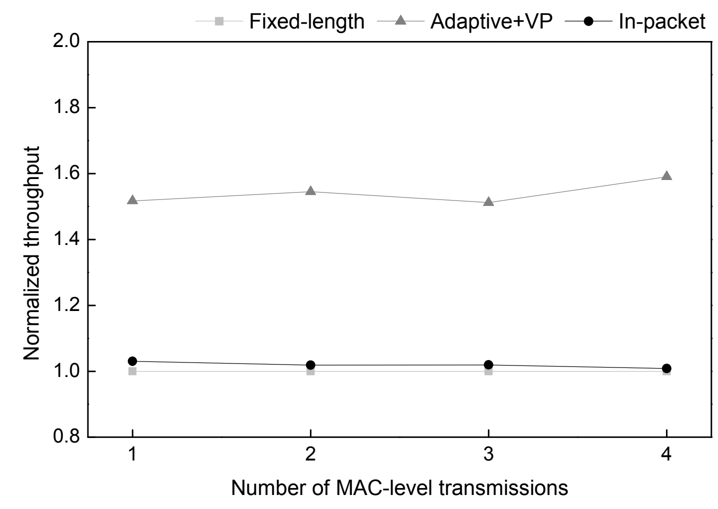
Fig. 7. Experimental set-up

ms for each data flow. The wireless network utilized an 802.11b access point (AP) operating in a distributed coordination function (DCF) mode. The receiver was arranged in clear line of sight (LoS) of the AP and generated wireless error-induced packet losses in accordance with a Gilbert channel model. A data transmission system was constructed in Linux 2.6.36 using a DCCP/TFRC module as a transport-layer rate control scheme. To support the partial checksum coverage feature in the link layer, the receiver was fitted with a wireless adapter based on the Atheros AR5BXB61 chipset [22]. The Linux driver for the chipset was modified from the Multiband Atheros Driver for WiFi (MADWiFi) [23]. During the data transmission, background traffic was generated by two TCP-based connections. To evaluate the fairness of the DCCP flow, the application packets were assigned the same size as those of the TCP background flows, i.e. 1000 bytes.

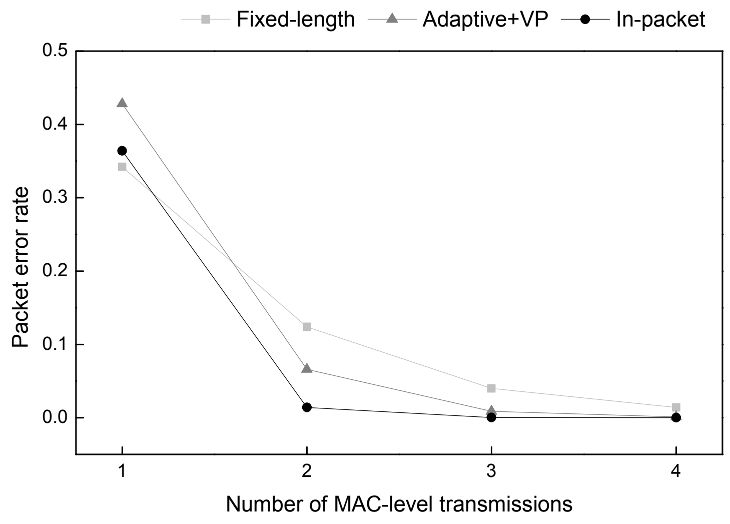
In the experiments, the performance of the proposed in-packet segmentation scheme was compared with that of the conventional fixed-length packetization scheme and the adaptive packetization scheme, respectively. Note that in the following discussions, packets which enter the DCCP/TFRC module are referred to as “application packets” while packets which are transmitted by the DCCP/TFRC module are referred to as “TFRC packets”. In the fixed-length packetization scheme, the TFRC packets are transmitted without any modification to the application packet size, whereas in the adaptive packetization scheme, the application packets are packetized into TFRC packets with an adaptive packet size utilizing the approach presented in Section 4.2 and the “virtual packets” mechanism is then employed to compensate for the TFRC throughput bias [19]. It is noted that the partial checksum coverage function is enabled for all three compared schemes. For the fixed-length packetization scheme and the adaptive packetization scheme, only the last received packet among multiple MAC-level transmissions is considered for performance evaluation purpose.

Figures 8(a) and 8(b) compare the throughput performance and packet error rate, respectively, of the three packetization schemes for various values of the maximum MAC-layer transmission (M) parameter. Note that in every case, the bit error rate and burst bit error length are assigned constant values of 10^{-3} and 20, respectively. In the experiments, the throughput of the schemes is normalized to TCP throughput and the packet error rate is estimated for the application packets. Overall, Figure 8 shows that for $M \geq 2$, the fixed-length packetization scheme has a fairer normalized throughput but a higher packet corruption rate than the other two schemes. In addition, it is seen that the proposed in-packet segmentation scheme not only achieves a reasonably fair normalized throughput, but also has a lower packet corruption rate than the other two schemes since the segment recovery technique enables the efficient recovery of multiple corrupted segments within a single retransmission. Finally, it is noted that due to the throughput bias induced by transporting small packets, the adaptive packetization scheme with the VP mechanism has a poorer throughput fairness than the conventional fixed-length packetization scheme, but achieves a lower packet corruption rate. The results presented in Figure 8(b) also show that the adaptive packetization scheme results in a higher packet corruption rate than the other two schemes for M=1. This result arises because the adaptive packetization scheme attaches a general network header to the head of each TFRC transport packet and therefore increases the likelihood of application packets comprising several small TFRC packets being corrupted.

Figures 9(a) and 9(b) compare the normalized throughput and packet error rate, respectively, of the three packetization schemes at various values of the bit error rate. Note that in each case, the burst bit error length and the maximum number of MAC-level transmissions are specified as 20 and 2, respectively. It can be seen in Figure 9(a) that the normalized throughput of the adaptive packetization scheme increases with an increasing bit error rate since the scheme uses a small packet size to transmit TFRC packets over a high bit-error-rate channel. In addition, it is observed that the normalized throughput of the proposed in-packet segmentation scheme is very similar to that of the fixed-length packetization scheme at all values of the bit error rate and remains approximately constant as the bit error rate is increased. The results presented in Figure 9(b) show that the in-packet segmentation scheme results in a similar packet corruption rate as the adaptive packetization scheme at all values of the bit error rate in the range $10^{-6} \sim 10^{-4}$, but achieves a significantly improved performance at a bit error rate



(a)

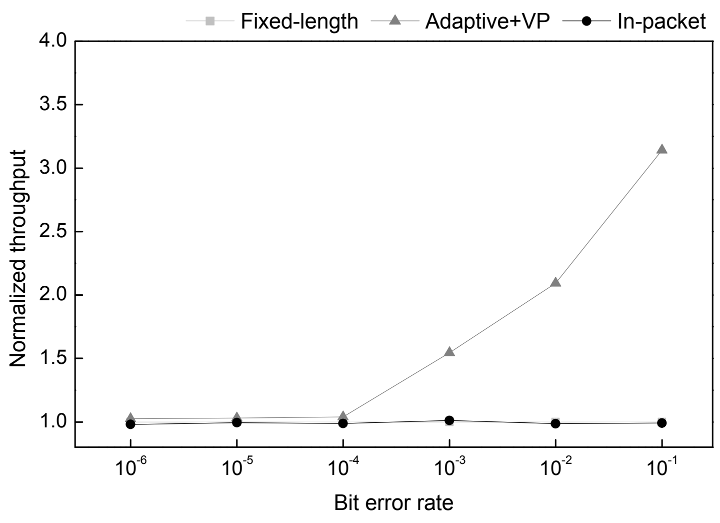


(b)

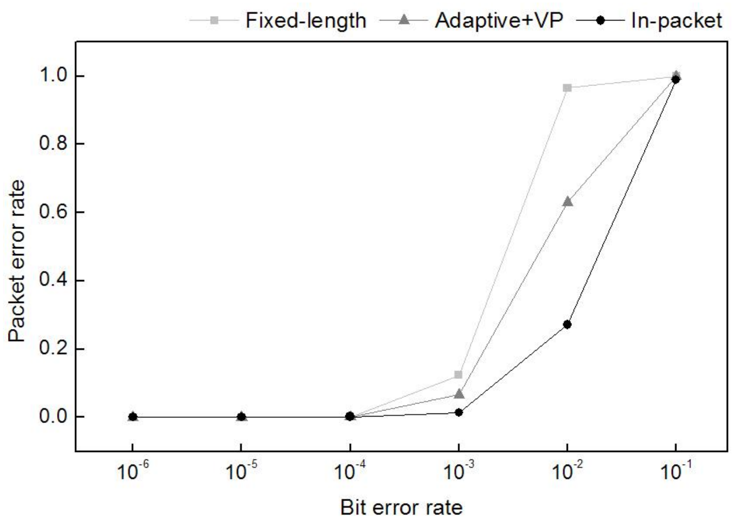
Fig. 8. Performance comparison of three packetization schemes for various maximum MAC-level transmission limit (*M*): (a) Normalized throughput; (b) Packet error rate

of 10^{-2}. It is noted that for a bit error rate of 10^{-1}, all the packets are lost during the wireless transmission irre spective of the packetization scheme applied. Figure 10 shows the performance results for the varied burst bit error length as the bit error rate and the number of maximum MAC-level transmissions are fixed to 10^{-3} and 2, respectively. While the burst loss pattern is considered in this experiment, the larger burst bit error length produces the lower packet error rate since more bit errors easily aggregate into the packet for a long burst bit error length with the fixed bit error rate. From Figure 10 (a), it can be seen that as the burst bit error length decreases, the adaptive packetization scheme has an increased normalized throughput since it uses smaller TFRC packets in response to higher packet error rate. In Figure 10 (b), as the burst bit error length is below 40 bits, the in-packet segmentation scheme outperforms the other two schemes and as the burst bit error length exceeds 40 bits, all three schemes perform alike due to the very low packet error rate.

In general, the results presented in Figures 8~10 show that the proposed in-packet segmentation scheme achieves a reasonably fair TFRC throughput and results in a similar or lower packet error rate than the adaptive packetization scheme under various wireless network conditions.



(a)

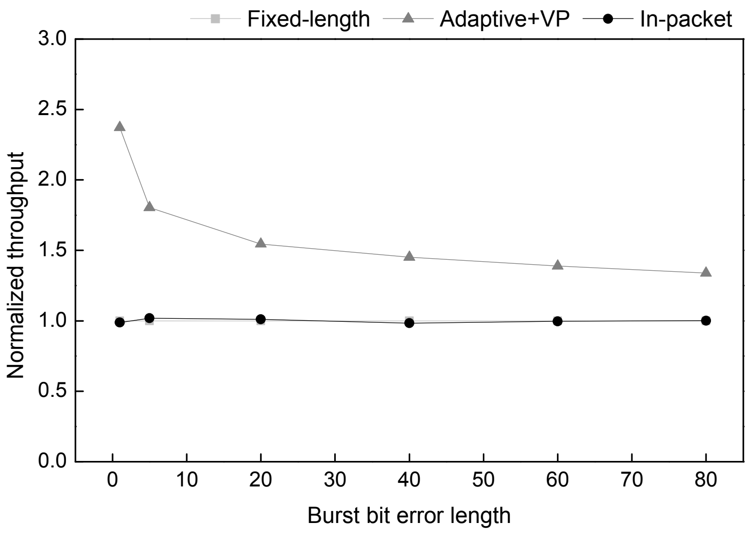


(b)

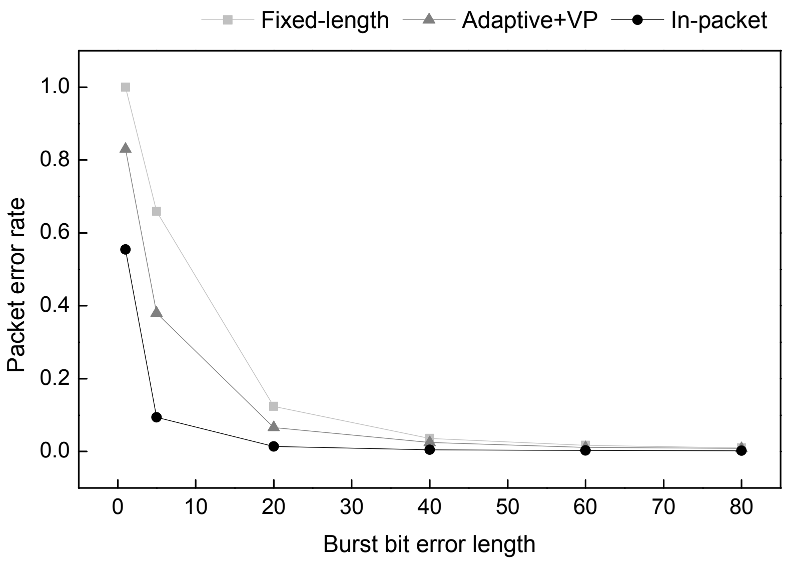
Fig. 9. Performance comparison of three packetization schemes for various bit error rates: (a) Normalized throughput; (b) Packet error rate

6 Conclusions

In this paper, we enhance DCCP protocol over ARQ-based wireless channels and propose an in-packet seg mentation scheme to achieve the efficient packetization of DCCP flows. Based on the DCCP enhancement, the in-packet segmentation scheme combined with the MAC-level ARQ protocol ensures TFRC throughput fairness and provides a robust data transmission performance through its use of an efficient segment recovery mechanism based on a partial checksum coverage technique. The experimental results have shown that the proposed scheme achieves a reasonably fair TFRC throughput and yields a similar or lower packet error rate than that obtained using existing adaptive packetization schemes. Thus, the in-packet segmentation scheme presented in this study provides an ideal packetization solution for the TCP-friendly streaming flows over time-varying lossy channels such as those in the wireless transmission environment. Based on the virtual segment technique, the future studies include both the media-aware segmentation scheme and the segment-level channel coding mechanism to improve the overall transport quality for DCCP applications.



(a)



(b)

Fig. 10. Performance comparison of three packetization schemes for various burst bit error length: (a) Normalized throughput; (b) Packet error rate

References

[1] E. Kohler, M. Handley, S. Floyd, J. Padhye, "Datagram Congestion Control Protocol (DCCP)," *RFC 4340*, March 2006.

[2] S. Floyd, K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, Vol. 7, No. 4, pp. 458-472, 1999.

[3] S. Floyd, E. Kohler, "Profile for DCCP Congestion Control ID 2: TCP-like Congestion Control," *RFC 4341*, March 2006.

[4] M. Handley, S. Floyd, J. Padhye, J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," *RFC 3448*,

January 2003.

[5] E. Kohler, S. Floyd, J. Padhye, "Profile for DCCP Congestion Control ID 3: TFRC Congestion Control," *RFC 4342*, March 2006.

[6] M.A. Azad, R. Mahmood, T. Mehmood, "A Comparative Analysis of DCCP Variants (CCID2, CCID3), TCP and UDP for MPEG4 Video Applications," in *Proceedings of 2009 International Conference on Information and Communication Technologies (ICICT 2009)*, pp.40-45, IEEE Press, 2009.

[7] I.S. Chowdhury, J. Lahiry, K.C. Rahman, S.F. Hasan "Performance Analysis of Datagram Congestion Control Protocol (DCCP)," *International Journal of Computer Theory and Engineering*, Vol. 3, No. 5, pp. 632-637, October 2011.

[8] S. Floyd, M. Handley, J. Padhye, J. Widmer, "Equation-Based Congestion Control for Unicast Applications," *ACM Computer Communication Review*, Vol. 30, No. 4, pp. 43-56, October 2000.

[9] T. Phelan, "Strategies for Streaming Media Applications Using TFRC," *IETF Internet draft*, July 2007.

[10] C.H. Shih, J.U. Wang, C.K. Shieh, W.S. Hwang, "An Integrated Rate Control Scheme for TCP-friendly MPEG-4 Video Transmission," *Signal Processing: Image Communication*, Vol. 23, No. 2, pp. 101-115, Feb. 2008.

[11] MPEG-4 Video Verification Model v18.0, Coding of Moving Pictures and Audio N3908, ISO/IEC JTC1/SC29/WG11, Jan. 2001.

[12] L.A. Larzon, M. Degermark, S. Pink, L.E. Jonsson, G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)," *RFC 3828*, July 2004.

[13] S.A. Khayam, S. Karande, H. Radha, D. Loguinov, "Performance Analysis and Modeling of Errors and Losses over 802.11b LANs for High-bit-rate Real-time Multimedia," *Signal Processing: Image Communication*, Vol. 18, No. 7, pp.575-595, Aug. 2003.

[14] S. Choudhury, J.D. Gibson, “Payload Length and Rate Adaptation for Multimedia Communications in Wireless LANs,” *IEEE Journal on Selected Areas in Communications*, Vol. 25, No. 4, pp. 796-807, May 2007.

[15] B. Gorkemli, M. R. Civanlar, "SVC Coded Video Streaming over DCCP," in *Proceedings of* 8th *IEEE International Symposium on Multimedia (ISM 2006)*, pp. 437-441, IEE Press, Dec. 2006.

[16] Y.C. Lai, C.N. Lai, "DCCP partial reliability extension with sequence number compensation," *Computer Networks*, Vol. 52, No. 16, pp. 3085-3100, Nov. 2008.

[17] M. Schier, M. Welzl, "Content-aware selective reliability for DCCP video streaming," in *Proceedings of International Conference on Multimedia Computing and Information Technology (MCIT 2010)*, pp.53-56, IEEE Press, March 2010.

[18] S. Floyd, E. Kohler, "TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant," *RFC 4828*, April 2007.

[19] J. Widmer, C. Boutremans, J.Y. Le Boudec, "End-to-end Congestion Control for TCP-Friendly Flows with Variable Packet Size," *ACM SIGCOMM Computer Communications Review*, Vol. 34, No. 2, pp. 137-151, Apr. 2004.

[20] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-time Applications," *RFC 3550*, July 2003.

[21] B. Gorkemli, M.O. Sunay, A.M. Tekalp, "Video Streaming over Wireless DCCP," in *Proceedings of 2008 IEEE International Conference on Image Processing (ICIP 2008)*, pp. 2028-2031, IEEE Press, Oct. 2008.

[22] "ATHEROS Communications." [Online]. Available: http://www.atheros.com/pt/index.html/

[23] "MADWIFI." [Online]. Available: http://madwifi.org/