# A Rate Control Scheme for Adaptive Real-Time Applications in IP Networks With Lossy Links and Long Round Trip Times

Ian F. Akyildiz, *Fellow, IEEE*, Özgür B. Akan, *Member, IEEE*, and Giacomo Morabito

***Abstract*—Currently there is no control for real-time traffic sources in IP networks. This is a serious problem because real-time traffic can not only congest the network but can also cause unfairness and starvation of TCP traffic. However, it is not possible to apply current solutions for Internet to the networks with high bandwidth-delay products and high bit error rates. The channel errors may result in inaccurate congestion control decisions and unnecessary rate throttles leading to severe performance degradation. This problem is amplified in the links with high bandwidth-delay products, since the link is inefficiently utilized for a very long time until the unnecessary rate throttle is recovered. In this paper, a new Rate Control Scheme, RCS, is introduced for real-time interactive applications in networks with high bandwidth-delay products and high bit error rates. RCS is based on the concept of using dummy packets to probe the availability of network resources. Dummy packets are treated as low priority packets and consequently they do not affect the throughput of actual data traffic. Therefore, RCS requires all the routers in the connection path to support some priority policy. A new algorithm is also proposed to improve the robustness of the RCS to temporal signal loss conditions. The delay-bound considerations for real-time traffic sources using RCS rate control scheme are also investigated. Simulation experiments show that in environments with high bandwidth-delay products and high bit error rates, RCS achieves high throughput performance without penalizing TCP connections.**

***Index Terms*—Flow control, high bandwidth-delay products, high bit error rates, real-time protocols.**

## I. Introduction

REAL-TIME applications have strict requirements on end-to-end delay. For this reason the *Differentiated Service* and *Integrated Service* models have been proposed for Internet in recent years. Both of them try to guarantee lower bounds on the bandwidth allocated to each flow and, consequently, upper bounds on the end-to-end delay. Both Differentiated and Integrated Service models require a high amount of resources and, as a result, the services relying on them are expected to have high costs. On the other hand, there will be a large number of users interested in using real-time applications at a low cost. These users will share network resources without any reservation.

In a shared network, such as the Internet, all traffic flows are expected to be *good network citizens* or *TCP friendly* [20], i.e., their transmission rate can increase as long as the network is not congested, and must decrease immediately when the network is congested.

In recent years, much research has been conducted to develop rate control protocols for real-time multimedia applications in the terrestrial wireline networks [4], [8], [13], [14], [16], [19], [20], [23]. However, the major drawback of these solutions is that they are developed for wired links which are assumed to have negligible errors and hence these solutions cannot be directly applied to the links with high bit error rates and high bandwidth-delay products.

Packet losses are the only congestion sign in the current Internet. However, some links, such as wireless, and satellite links, are characterized by high link error rates and thus, packet loss can occur due to link errors with probability even higher than $10^{-2}$. If the source decreases its transmission rate when a packet loss occurs due to link errors, then the network efficiency decreases drastically, i.e., it can be lower than 20%. This problem is amplified by the long delays involved in most Internet communications. Moreover, even higher RTT values have been observed in Wireless Wide Area Networks (WWAN). A typical round-trip time (RTT) varies between few hundred milliseconds and seconds in cellular WWAN links due to the extensive physical layer processing, e.g., forward error correction (FEC), interleaving and transmission delays [15]. The access delay is much higher in satellite links, which have high propagation delay up to 270 ms [1].

Delay can also be high because of the high hop distance between the two end systems. In fact, each hop causes a new queuing and processing delay. The current average hop distance is about 16 and is expected to increase in the future [17].

Consequently, despite the existence of many proposed rate control schemes for real-time multimedia flows in the terrestrial wireline networks, there still exists a need for a new rate control scheme that can address the adverse effects of the high bit error rates and high bandwidth-delay products. In order to address this need, in this paper, a Rate Control Scheme (RCS) is proposed for real-time traffic in networks with high bandwidth-delay products and high bit error rates.

Manuscript received June 17, 2002; revised October 6, 2003 and April 22, 2004; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Campbell. This work was supported by NASA-Ames Research Center under Contract NAG2-1262. An earlier and shorter version of this work was published in the Proceedings of the IEEE INFOCOM, Anchorage, AK, April, 2001.

Ian F. Akyildiz is with the Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: ian@ece.gatech.edu).

Ö. B. Akan was with the Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. He is now with the Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara, Turkey 06531 (e-mail: akan@eee.metu.edu.tr).

G. Morabito is with the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, University of Catania at Enna, Catania, Italy 95124 (e-mail: giacomo.morabito@diit.unict.it).

Digital Object Identifier 10.1109/TNET.2005.850225

The remainder of the paper is organized as follows. In Section II, we extensively explore the current related work in the literature. In Section III, we introduce RCS and in Section IV we show its behavior in three different cases: First, when a packet loss occurs due to link errors; second, when a packet loss occurs due to network congestion; third, when a packet loss occurs due to temporary signal loss. Simulation results in Section V show that in case of networks with high bandwidth-delay products and high bit error rates, RCS improves the efficiency without penalizing TCP traffic. Finally, we conclude the paper in Section VI.

## II. Related Work

In this section, we report the related work provided in the literature; i.e., rate control schemes proposed for multimedia traffic, transport layer solutions proposed to address the performance degradation due to high link error rate, and the existing link probing mechanisms.

### A. Rate Control for Real-Time Traffic in the Internet

The need of rate control for multimedia flows in the Internet has been subject of much research in the past. Three approaches for rate control of real-time traffic in the Internet can be distinguished:

1) *TCP-like protocols:* TCP-like protocols [4], [8], [14], [19], [20], [23] follow AIMD data rate control, i.e., transmission rate is halved in case of packet loss. This causes severe performance degradation in the links with high bit error rates. Furthermore, the recovery from such unnecessary transmission rate drop takes a time period proportional to the RTT. Hence, the above problem is amplified in case of long propagation delays.
2) *Equation-based protocols:* The source uses a control equation based on the statistics of the RTT and packet loss probability to estimate the available bandwidth and set its transmission rate accordingly [13]. However, the existing equation-based rate control schemes cannot be applied to the links with high bit error and high propagation delays since the throughput equation in [18] models the steady-state TCP behavior over error-free wireline links. Therefore, it may result in inaccurate rate estimation and hence underutilization of the link resources.
3) *Explicit bandwidth notification-based protocols:* This approach [16] requires each router to continuously perform per flow buffer and bandwidth monitoring and to send explicit feedback. Furthermore, this method also does not consider link errors which may avoid the arrival of the required explicit feedback and lead to erroneous transmission rate estimation.

### B. Transport Protocols for Lossy Links

The problem of performance degradation due to link errors has been the subject of significant research efforts in recent years. We can distinguish three types of approaches:

1) *Modifying TCP:* In this approach, modifications are introduced in the end hosts and hence the end-to-end semantics are maintained [1], [2], [22]. However, these algorithms perform retransmissions to guarantee end-to-end reliability, which would lead to waste of wireless resources for the delivery of loss-tolerant, time-sensitive multimedia flows.
2) *Isolating the wireless link:* There exist solutions to address the same problem by isolating the error-prone link from the rest of the end-to-end connection path [3], [5]. However, local retransmissions to isolate link errors from the source result in waste of scarce wireless link resources since retransmissions are not necessary for the multimedia flows.
3) *Explicit Congestion Notification (ECN):* The router experiencing certain level of congestion marks the ECN bit of the IP header and forwards the packet [12]. The source receiving the marked packet takes appropriate congestion control measures. Such approach would help to minimize erroneous congestion detections caused by packet losses due to link errors. However, there is always the possibility of losing ECN marked packets due to link errors, which leads to inaccurate congestion control.

### C. Active Probing Protocols

Active probing approach has also been used to estimate the available bandwidth for resource reservation/adaptation and sending rate control in wireless networks [7], [6], [11], [22]. However, none of these proposed solutions use probe packets to emulate TCP behavior to achieve TCP-friendliness while performing rate control of multimedia flows.

## III. RCS: Rate Control Scheme

RCS is an end-to-end rate control scheme which uses AIMD [9] congestion control approach, in order to produce TCP-friendly traffic flows while maintaining high throughput performance in networks with high bandwidth-delay products and high bit error rates.

RCS runs on top of RTP/RTCP [21] and UDP and is mainly implemented at the source but also needs some functions at the destination. In fact, at the destination RCS layer sends back an acknowledgment (ACK) for any received packet, as suggested also in [20]. Note that these ACKs are used only for flow control as will be explained in the following. No retransmissions are performed. At the destination, RCS layer passes the received data packet to the decoder.

RCS protocol consists of four main algorithms specifically tailored to address the real-time multimedia transport in the networks with high bit error rates and high bandwidth-delay products.

In order to effectively handle the adverse effects of the high propagation delay, RCS contains *Initial* algorithm which is specifically tailored to capture the link resources in a fast and controlled manner in the links with high bandwidth-delay products. The details of the Initial State behavior are explained in Section III-B. The link probing technique exploited in the Initial State is based on the usage of low priority *dummy packets* [1]. The details of the dummy packets concept are given in Section III-A.

After the initial data transmission rate is set by the Initial State procedure, RCS source executes *Steady* algorithm which performs additive-increase rate control. In order to address the challenges due to high bit error rates, RCS source calls *Detected* algorithm in case of a packet loss. The details of the *Steady* and

*Detected* algorithm operations are presented in Sections III-C and III-D, respectively.

Furthermore, in order to address the possible temporal signal loss conditions, RCS incorporates *Backoff* algorithm into its protocol operation. The objective of the Backoff algorithm is to capture the link loss condition and hence minimize the performance degradation by avoiding unnecessary rate throttles. The Backoff algorithm operation is discussed in Section III-E.

### A. Dummy Packets

RCS is based on the use of dummy packets. Unlike the other existing link probing schemes discussed in Section II-C, dummy packets are low priority packets used by the source to probe the availability of network resources [1]. If a router on the connection path is congested, then it discards the IP packets carrying dummy packets first. Consequently, the transmission of dummy packets does not cause a decrease in the throughput of actual *data packets*. If the routers are not congested, then the dummy packets can reach the destination which sends the related acknowledgments. The ACKs for dummy packets are also carried by the low priority IP packets. The source interprets the ACKs for dummy packets as the evidence that there are unused resources in the network and accordingly, can increase its transmission rate. Note that dummy packets may produce some overhead, but we demonstrate in Section V that they use resources which otherwise would be unutilized.

The new scheme requires all routers in the connection path support some priority discipline. In fact, RCS injects dummy packets into the network regardless of the current traffic load. As a consequence, dummy packets may congest routers and affect data packet throughput if a router on the connection path does not apply any priority policy. Note that in traditional IP networks the IP *type of service* (TOS) can be used for this purpose. In fact, one of the eight bits of the TOS field in the IP header gives the priority level of the IP packet. Instead, more recent IP versions, e.g., IPv6, explicitly provide several priority levels. Therefore, since both dummy packets and their ACKs are encapsulated by low priority IP packets, RCS source and the receiver can distinguish both dummy packets and the ACKs for dummy packets from the normal data packets via TOS field in the IP packet header. Currently, most of the commercial routers, e.g., Cisco series 2500, 4500 and higher, already implement *priority-queuing* capability and use the TOS field of the IP packet header. The details of the various priority-queuing methods are beyond the scope of the paper.

Applications generating low priority traffic may be penalized by dummy packets even if priority is supported by routers. The transmission of dummy packets may cause congestions for low priority traffic. However, dummy packets are not continuously transmitted rather they are transmitted for short periods of time. Hence, those possible congestion situations last for a short period. In fact, dummy packets are transmitted only in two cases, i.e., in the beginning of a new connection (for a period of one RTT) and when a data packet is lost, in which case dummy packet transmission may harm low priority data traffic especially if the data packet loss is due to link error. However, we will show that the payback for this problem is a high increase of the throughput of high priority traffic.

As shown in Fig. 1, RCS source is a finite state machine model with four states: *Initial, Steady, Detected*, and *Backoff*. In the following we present the behavior of RCS source in each of the above states.

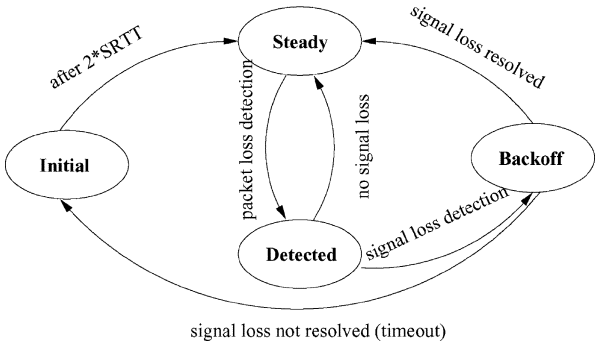


Fig. 1. Finite state machine model of the RCS source.

```
Initial()
    t_0 = t;
    t_1 = t_0 + SRTT;
    t_END = t + 2 · SRTT;
    IPG_Dummy = 1/S_Target;
    t_next_dummy = t_0 + IPG_Dummy;
    n_ACK = 0;
    while (t ≤ t_END)
        while (t ≤ t_1)
           while(t < t_next_dummy)
              if (DUMMY_ACK_ARRIVAL)
                 n_ACK = n_ACK + 1;
              end;
           end;
           send(DUMMY_PACKET);
           t_next_dummy = t_next_dummy + IPG_Dummy;
        end;
        if (DUMMY_ACK_ARRIVAL)
           n_ACK = n_ACK + 1;
        end;
    end;
    wdsn = −1;
    S = max(1, n_ACK)/SRTT;
    state=Steady;
end.
```

Fig. 2. `Initial()` Algorithm.

### B. Initial State Behavior

In the beginning of a new connection, the source must set the initial transmission rate value, $S_{\text{Initial}}$. Let $S_{\text{Available}}$ be the transmission rate sustainable by the network. The choice of $S_{\text{Initial}}$ is important, because if $S_{\text{Initial}} \ll S_{\text{Available}}$, then the new connection will cause network congestion, otherwise, resource utilization will be very low and will stay low for a time interval proportional to the bandwidth-delay product.

In order to effectively handle the adverse effects of the high propagation delay and hence efficiently set the transmission rate value $S$, RCS incorporates the `Initial()` algorithm given in Fig. 2 which is the new procedure specifically tailored to capture the link resources in a fast and controlled manner in links with high bandwidth-delay products.

RCS starts a new connection in the Initial state and remains there for a time interval, $t_{\text{Initial}}$, equal to two times the estimated round trip time, SRTT, i.e., $(t_{\text{Initial}} = 2 \cdot \text{SRTT})$. Hence, the objective of Initial state phase is to set the initial transmission rate as early as $t = 2 \cdot \text{RTT}$ without leading to any congestion. RCS maintains an SRTT value as in case of TCP. Although there are different methods to compute SRTT, note that SRTT is only

utilized as a reference timescale here, thus, RCS performance is independent of the SRTT selection.

Let $t$ represent the current system time and $t_0$ be the initial time instant. The Initial phase lasts for two times SRTT, as a result, the `Initial()` algorithm will terminate at the time $t_{\rm END}$, given by

$$t_{\rm END} = t_0 + 2 \cdot {\rm SRTT}. \tag{1}$$

Let $S_{\rm Target}$ represent the value of the data transmission rate needed to transmit the real-time stream with the highest quality. For example:

- If layered encoding is used, then $S_{\rm Target}$ is the transmission rate needed to transmit all the layers of the encoded stream.
- If adaptive encoding is used, then $S_{\rm Target}$ is the transmission rate needed to transmit the real-time stream encoded with the highest definition.

During $t_0 \leq t \leq (t_1 = t_0 + {\rm SRTT})$, RCS source sends dummy packets (`send(DUMMY_PACKET)`) at rate $S_{\rm Target}$, i.e., the inter-transmission time is $({\rm IPG}_{\rm Dummy} = 1/S_{\rm Target})$. RCS uses the variable $t_{\rm next\_dummy}$ to indicate the time when the next dummy packet has to be sent. After each dummy packet transmission, $t_{\rm next\_dummy}$ is updated as shown in Fig. 2.

If there are sufficient bandwidth resources available at the routers along the path, the dummy packets transmitted in $t < {\rm RTT}$ are received and ACKed back by the receiver as explained in Section III-A. The source interprets the ACKs for dummy packets as the evidence that there are unused resources in the network. Therefore, RCS source counts the number, $n_{\rm ACK}$, of ACKs received for dummy packets.

Consequently, at the end of the Initial state, i.e., $t = 2 \cdot {\rm RTT}$, RCS source sets the data transmission rate $S$ by using the network resource availability information captured by the dummy packets, i.e.,

$$S = \frac{\max\{1, n_{\rm ACK}\}}{\rm SRTT}. \tag{2}$$

Furthermore, before leaving the Initial State, RCS sets the variable $wdsn = -1$. As it is clarified in the Section III-D, RCS uses the variable $wdsn$ in order to preserve TCP-friendliness.

### C. Steady State Behavior

In the Steady state, RCS source assumes that the network is not congested. Thus, according to the additive-increase scheme [9], it increases its transmission rate in a step-like fashion periodically. Moreover, upon receiving an ACK for a dummy packet, the RCS source checks the value of the variable $wdsn$. We use the variable $wdsn$ in order to match RCS source behavior with TCP behavior when the network is congested [1]. If $wdsn$ is higher than zero, then RCS source decreases $wdsn$ by one, i.e., $(wdsn = wdsn - 1)$. Otherwise, RCS source increases its transmission rate by one packet per estimated round trip time, SRTT. The details of the determination of the variable $wdsn$ and how it assures TCP-friendliness are explained along with the Detected state behavior in Section III-D. RCS source leaves the Steady state for the Detected state when a data packet loss is detected. Note that RCS source uses the same mechanism of TCP Reno to detect data packet losses.

```
Steady()
    END=0;
    t_0 = t;
    t_next_data = t_0;
    t_next_increase = t_0 + SRTT;
    while (END == 0)
        if (PACKET_LOSS_DETECTION)
            END=1;
        end;
        if (t ≥ t_next_data)
            send(DATA_PACKET);
            t_next_data = t_next_data + IPG;
        end;
        if (t ≥ t_next_increase)
            S = min(S + 1/SRTT, S_Target);
            IPG = 1/S;
        if (DUMMY_ACK_ARRIVAL)
            if (wdsn == 0)
                S = min(S + 1/SRTT, S_Target);
                IPG = 1/S;
            else
                wdsn = wdsn − 1;
            end;
        end;
    end;
    state=Detected;
end.
```

Fig. 3. `Steady()` Algorithm.

During the Steady phase, RCS source executes the `Steady()` algorithm shown in Fig. 3. The algorithm uses the following variables:

- $t_0$: It gives the time instant when the current Steady phase started.
- $t_{\rm next\_data}$: It is the time instant when the next data packet has to be sent. When the current time, $t$, is higher than or equal to $t_{\rm next\_data}$, a data packet is sent (`send(DATA_PACKET)`), and $t_{\rm next\_data}$ is updated.
- IPG: It is the time interval between two successive data packet transmissions and is given by ${\rm IPG} = 1/S$, where $S$ is data transmission rate.
- $t_{\rm next\_increase}$: It is the time instant when the transmission rate, $S$, must be increased. According to the additive increase scheme [9], $S$ is increased periodically in a step-like fashion. In order to match the behavior of RCS source with the behavior of TCP, the period is SRTT and the step height is $1/{\rm SRTT}$. However, the transmission rate, $S$, never exceeds the value $S_{\rm Target}$. As a consequence, if $t \geq t_{\rm next\_increase}$, the transmission rate is updated as follows:

  $$S = \min\{S + 1/{\rm SRTT}, S_{\rm Target}\} \tag{3}$$

  and $t_{\rm next\_increase}$ is updated

  $$t_{\rm next\_increase} = t_{\rm next\_increase} + {\rm SRTT}. \tag{4}$$

- $wdsn$: When the ACK for a dummy segment is received, i.e., `DUMMY_ACK_ARRIVAL`, RCS source checks the value of $wdsn$. If $wdsn = 0$, then the transmission rate is increased as in (3), otherwise $wdsn$ is decreased by one, i.e., $wdsn = wdsn - 1$.

### D. Detected State Behavior

In order to address the challenges due to high bit error rates, RCS source incorporates the *Detected* algorithm. RCS source enters the Detected state when a data packet loss is detected.

```
Detected()
    t_0 = t;
    t_END = t_0 + SRTT;
    S = S/2;
    IPG = 1/S;
    t_next_data = t_0;
    wdsn = SRTT · S;
    ack_received = false;
    while (t ≤ t_END)
        if (t ≥ t_next_data)
            send(DATA_PACKET);
        if (t ≥ t_next_data + IPG/3)
            send(DUMMY_PACKET);
        if (t ≥ t_next_data + 2 · IPG/3)
            send(DUMMY_PACKET);
        t_next_data = t_next_data + IPG;
        if (ACK_ARRIVAL)
            ack_received=true;
    end;
    if (ack_received ==true)
        state=Steady;
    else
        state=Backoff;
end.
```

Fig. 4. `Detected()` Algorithm.

In the Detected state, RCS source executes the `Detected()` algorithm shown in Fig. 4. Since the reason for the packet loss is unknown to the source at the beginning, it is safer to decrease the data rate to preserve TCP-friendly behavior. As a result, RCS source maintains the TCP conservative assumption that all packet losses are due to network congestion and, accordingly, halves its data transmission rate, $S$ upon entering Detected state. The Detected phase lasts for a time interval equal to the estimated round trip time, SRTT, thus, it finishes at time ($t_{\rm END} = t_0 + {\rm SRTT}$), where $t_0$ is the time instant when the Detected algorithm is initiated. At the end of the Detected phase, RCS source goes back to the Steady or Backoff state as shown in Fig. 1.

Furthermore, the sender also transmits dummy packets in order to differentiate packet losses due to congestion and link errors by probing the actual availability of network resources. In the Detected phase, data packets are sent with rate $S$ and two dummy packets are transmitted for each data packet. Packet transmissions are uniformly distributed, thus, the time interval between two successive transmissions is ${\rm IPG}/3$. The reason for transmitting two dummy packets for each data packet is explained below.

The ACKs for the dummy packets transmitted in the Detected state are received in the Steady state. Let $S_0$ be the data transmission rate when the packet loss occurs. If the packet loss is due to congestion, then the congested router can serve $S_0 \cdot {\rm SRTT}$ packets per RTT, approximately. As a result, the network will accommodate $S_0 \cdot {\rm SRTT}/2$ data packets, which have high priority, and $S_0 \cdot {\rm SRTT}/2$ dummy packets out of the $S_0 \cdot {\rm SRTT}$ dummy packets transmitted during the Detected state. Therefore, the sender must not increase its transmission rate $S$ when it receives the first $S_0 \cdot {\rm SRTT}/2$ ACKs for dummy packets. In fact, these ACKs cannot be considered as the sign that the loss was due to link errors, i.e., not due to network congestion. Accordingly, $wdsn$ is set to $S_0 \cdot {\rm SRTT}/2$ in Detected state as shown in Fig. 4. As explained in Section III-C, the sender checks the value of $wdsn$ before increasing the data rate upon reception of an ACK for a dummy packet. Hence, this assures TCP-friendliness in case of congestion by preventing the sender to increase its transmission rate when the first $S_0 \cdot {\rm SRTT}/2$ ACKs for dummy packets are received during the Steady state.

After receiving $S_0 \cdot {\rm SRTT}/2$ ACKs for dummy segments, the sender increases its transmission rate by $1/{\rm SRTT}$ each time it receives an ACK for a dummy segment. As a result, if all of the dummy segments transmitted in the Detected state are ACKed to the sender, i.e., the packet loss is due to link error, then the data transmission rate $S$ reaches the value it had before the packet loss was detected, i.e., $S = S_0$. This behavior is further traced in Section IV.

In summary, with the help of the ACKs received for the dummy packets transmitted in Detected state, the source can accurately distinguish congestion and link error related packet losses and hence recover from the rate throttle in case of link errors in the Steady state. If the packet loss is due to congestion, then $wdsn$ variable assures that RCS maintains TCP-friendliness and does not increase its transmission rate in the Steady state with the reception of ACKs for dummy packets. If the data packet loss is due to link errors, i.e., the network is not congested, then these dummy packets are acknowledged and the data transmission rate, $S$, is recovered accordingly.

Moreover, it is also likely to experience intermittent link blockages and signal losses in land mobile satellite communication systems due to handoff or signal fading by environmental obstructions such as tunnels, bridges, mountains, and weather patterns such as rainstorms [24]. In such blockage periods, data cannot be successfully transmitted to the receiver and hence performance severely degrades. However, it is essential to make sure that as soon as the blockage period is over the delivery of the multimedia resumes at the highest possible rate, hence minimizing disruption to the on-going connections. In order to achieve this, RCS source incorporates Backoff algorithm whose details are presented in Section III-E.

Therefore, RCS source waits for an ACK during the Detected state to assess the actual reason for the packet loss event. If the packet loss is due to the random link error, then RCS source receives an ACK during the SRTT period of Detected state. If the packet loss is due to congestion, then all TCP friendly protocol sources along the bottleneck would perform rate throttle for congestion resolution in at most one SRTT. Hence, if no ACK is received until the end of one SRTT, it can be inferred as an indication of temporary signal loss instead of the congestion without losing accuracy. In this case, turning back to the Steady state at the end of Detected state is not a good idea. If signal loss is not resolved yet by the end of Detected phase and the source goes into Steady state, it detects another packet loss and turns back to Detected state halving its transmission rate $S$ once again.

This consecutive rate decrease problem can severely degrade the throughput efficiency. In order to avoid this problem, RCS source waits for an ACK during the Detected phase. If any ACK is received, this means that signal loss is resolved and next state is set to Steady state, i.e., ${\rm state} = {\rm Steady}$. If no ACK is received during the Detected phase, then the RCS source does not go back to Steady state, instead it goes to Backoff state, i.e., ${\rm state} = {\rm Backoff}$ as shown in Fig. 1.

### *E. Backoff State Behavior*

In order to prevent unnecessary rate decrease until temporary signal loss is resolved, RCS source backoffs at the end of the Detected phase, i.e., holds its current transmission rate $S$,

```
Backoff()
    t_0 = t;
    IPG = 1/S;
    wdsn=0;
    ack_received = false;
    t_next_data = t_0;
    while (ack_received == false)
        if (t ≥ t_next_data)
            send(DATA_PACKET);
        if (t ≥ t_next_data + IPG/2)
            send(DUMMY_PACKET);
        t_next_data = t_next_data + IPG;
        if (ACK_ARRIVAL)
            ack_received=true;
    end;
    state=Steady;
end.
```

Fig. 5. `Backoff()` Algorithm.

if no ACK is received in that phase. Hence, RCS source enters *Backoff* state and calls `Backoff()` algorithm as shown in Fig. 5. In this phase, RCS source sends both dummy and data packets with the same transmission rate $S$. RCS source remains in the Backoff state until it receives any ACK for dummy or data packets. At the end of the Backoff phase, RCS source goes to the Steady state as shown in Fig. 1.

In the Backoff state, RCS source executes the `Backoff()` algorithm shown in Fig. 5. The Backoff phase starts at $t_0$ and lasts until any ACK is received. During the algorithm execution,

- The transmission rate, $S$, and IPG are kept constant.
- The variable $wdsn$ is set to zero $wdsn = 0$.

During Backoff phase, RCS source stops halving its transmission rate $S$ and waits for an ACK (`ACK_ARRIVAL`) informing that signal is back. At the same time, RCS source sends data and dummy packets with equal transmission rate $S$. Once RCS source receives any ACK for either data or dummy packet, it goes into Steady state as in Fig. 1. If no ACK is received for a certain timeout period, $T_B$, the source timeouts and moves back to Initial State, as shown in Fig. 1.

The reason for continuing transmission of data packets in Backoff state is two-fold. Since the application is real-time multimedia, which is loss-tolerant to a certain extent and has strict time-deadlines, it is better to keep transmission going on instead of buffering the data until the signal is back. Secondly, while the blockage situation is inferred in the Detected state, it is also possible that the signal may be back during the Backoff state. Hence, instead of waiting until a reception of an ACK ending Backoff state, transmitting loss-tolerant time-sensitive multimedia packets can increase the received service quality.

Furthermore, RCS source sends one dummy packet for each data packet transmitted in Backoff state. The reason for transmitting dummy packets in Backoff state is as follows. Since $wdsn$ is set to 0 as shown in Fig. 5, in the Steady state the source increases its transmission rate $S$ by $1/\mathrm{SRTT}$ each time it receives an ACK for the dummy packets transmitted in this phase. Hence, in the following Steady state, the RCS source can increase its transmission rate $S$ immediately with the help of the dummy packets sent in the last SRTT part of the Backoff phase.

### *F. Protocol Architecture for Adaptive Real-Time Applications*

RCS evaluates the source transmission rate which must not be exceeded to be TCP-friendly. To achieve this goal, RCS must be inserted into an appropriate protocol stack, which is the focus of this section.

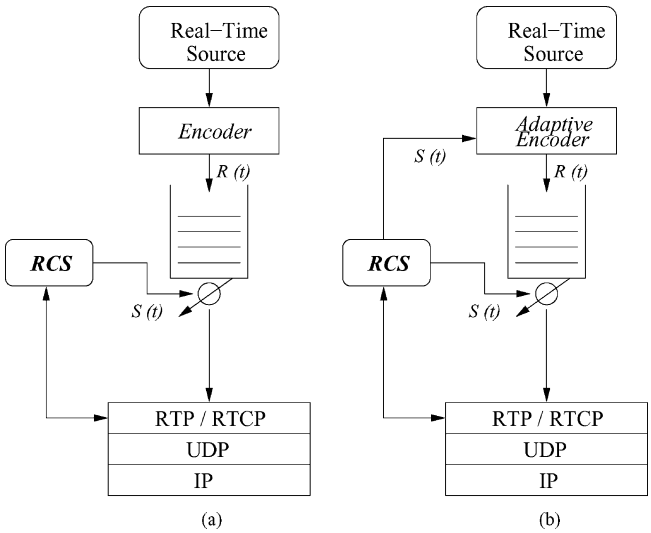


Fig. 6. TCP-Friendly architectures (a) without Adaptive Encoding and (b) with Adaptive Encoding.

Consider the system in Fig. 6(a). RCS is responsible of calculating at any time, $t$, the transmission rate, $S(t)$, that the traffic source should respect in order to be TCP-friendly. Since the output rate from the encoder, $R(t)$, is generally different from $S(t)$, i.e., $R(t) \neq S(t)$, a buffer with controlled output rate is inserted between the encoder and the RTP/RTCP layer. Accordingly, the transmission rate is equal to $S(t)$ and the source is TCP-Friendly. However, the above buffer introduces further policing delay, $d_\pi$, such that:

- The delay $d_\pi$ is low when the network is not congested and $S(t)$ is high.
- The delay $d_\pi$ is high when the network is congested and $S(t)$ is low. Moreover, some information may get lost due to buffer overflows.

As a consequence, buffering results in increased delay and delay jitter.

Real-time applications are very sensitive to delay and delay jitter thus, when $S(t)$ is low, it is better to decrease the quality of encoding rather than incur in further delays. As a consequence, we use encoders with feedback as shown in Fig. 6(b): RCS provides the encoder with $S(t)$ as input and the encoder adapts its parameters in such a way that $R(t)$ is as close to $S(t)$ as possible. In the ideal case, at any time $t$, $R(t) = S(t)$ and as a consequence the buffer size, $k(t)$, should be equal to zero as well as the delay $d(t)$, i.e., $k(t) = 0$ and $d(t) = 0$. However, real adaptive encoders are not able to comply exactly with $S(t)$, i.e., in general $S(t)$ and $R(t)$ are different. As an example, in Fig. 7 we show $R(t)$ and $S(t)$ obtained using adaptive video encoding.

The differences between $R(t)$ and $S(t)$ as shown in Fig. 7 result in a buffer whose size is given in the upper plot of Fig. 8 versus time. In the bottom plot of Fig. 8 we show the delay, $d(t)$, versus time. The values of $S(t)$ have been obtained by simulating RCS in the environment which will be described in Section V-A, whereas the $R(t)$ is the experimental output rate of a MSSG MPEG-2 video encoder when the target source rate is $S(t)$. Both the plots in Fig. 8, were obtained considering that

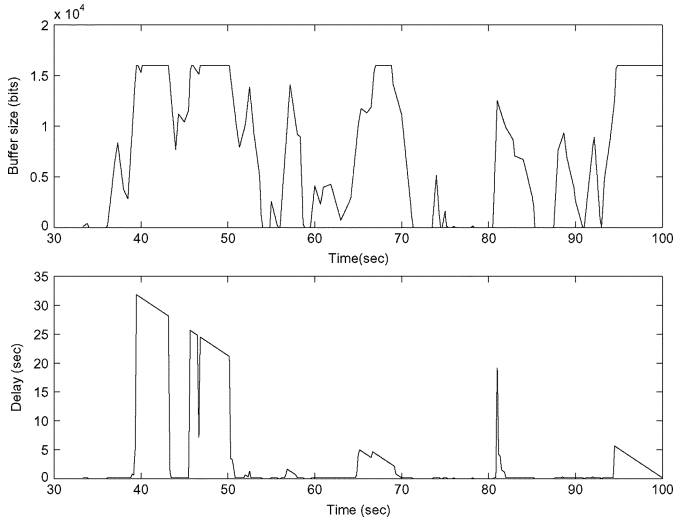


Fig. 7. Behavior of $R(t)$ and $S(t)$ with Adaptive Encoding.

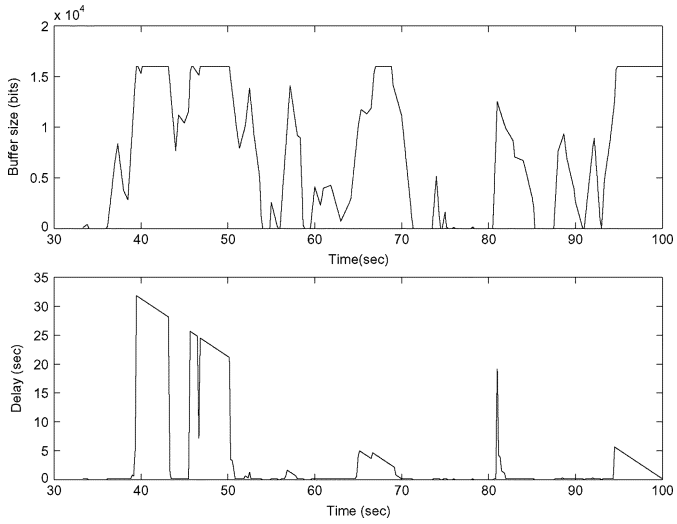


Fig. 8. Buffer size $k(t)$ and delay $d(t)$ with Adaptive Encoding.

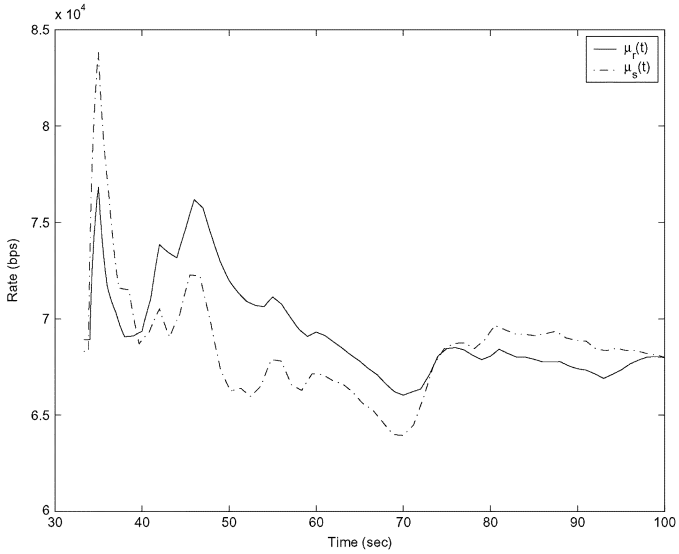


Fig. 9. Average rates $\mu_R(t)$ and $\mu_S(t)$ with Adaptive Encoding.

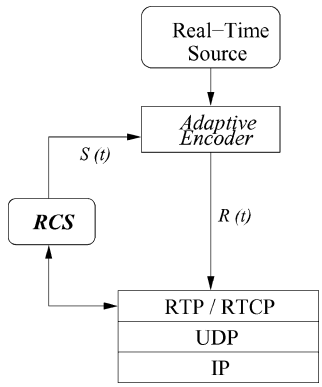


Fig. 10. Proposed TCP-Friendly architecture.

the maximum size of the buffer is 16 kb. Note that the delay can be higher than 30 s.

Let us define $\mu_R(t)$ and $\mu_S(t)$ as follows:

$$\mu_R(t) = \frac{1}{t} \cdot \int_0^t R(\tau)\, d\tau \tag{5}$$

$$\mu_S(t) = \frac{1}{t} \cdot \int_0^t S(\tau)\, d\tau. \tag{6}$$

Note that $\mu_R(t)$ and $\mu_S(t)$ are the average values of $R(t)$ and $S(t)$ in the time interval $[0, t]$. In Fig. 9 we show $\mu_R(t)$ and $\mu_S(t)$ achieved in the case given in Fig. 7. Observe that when $t$ is high, $\mu_R(t) \approx \mu_S(t)$. As a result, if a source generating traffic with rate $S(t)$ is TCP-friendly, a source generating traffic with rate $R(t)$ is TCP-friendly as well. From what we said before, we remove the buffer with controlled rate as shown in Fig. 10. Using such an architecture we can guarantee TCP-friendliness without introducing further delays.

## IV. RCS Behavior

In this section we show how the `Detected()`, `Steady()` and `Backoff()` algorithms cooperate when a data packet loss occurs due to link errors, congestion, and temporal signal loss in Sections IV-A, IV-B, and IV-C, respectively. In all cases, the RCS source is initially assumed to be at the Steady State at $t = t_0$. At $t = t_0$, a packet loss is detected and the source moves to Detected state as explained in Section III-D.

### A. Packet Loss Due to Link Errors

In this subsection, we describe the behavior of RCS on the consequence of a packet loss due to a link error. The detailed trace of RCS protocol operation in case of a packet loss due to link errors is presented as follows.

Let $t$ be time instance in Fig. 11.

- $t = t_0$
  $(S = S_0, \text{state} = \text{Steady})$.
  Suppose that the data transmission rate is $S_0$ at time $t_0$.
- $t_0 < t \leq t_1$ (where $t_1 = t_0 + \text{SRTT}$)
  $(S = S_0/2, wdsn = \text{SRTT} \cdot S_0/2, \text{state} = \text{Detected})$.
  Suppose at time $t_0$ the source detects a packet loss. The source enters the Detected state, halves its transmission rate, i.e., $S = S_0/2$, and sets $wdsn$ to $(wdsn_0 = \text{SRTT} \cdot S_0/2)$. Moreover, it transmits $(\text{SRTT} \cdot S_0)$ dummy packets with rate equal to $S_0$ as explained in Section III-D.

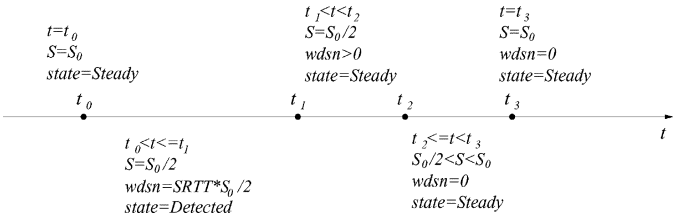


Fig. 11. RCS behavior when a packet loss occurs due to link errors.

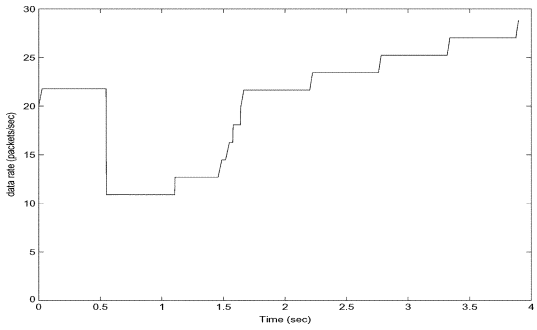


Fig. 12. Transmission rate when a packet loss is due to link errors.

- $t_1 < t < t_2$ (where $t_2 = t_1 + 0.5 \cdot \mathrm{SRTT}$)
  $(S = S_0/2, wdsn > 0, \mathrm{state} = \mathrm{Steady})$.
  At time $t = t_1$, the RCS source returns to the Steady state and starts to receive the ACKs for the dummy packets transmitted in the time interval $[t_0, t_1]$. If the network is not congested and thus, dummy packets are not lost, then the number of ACKs for dummy packets received in time interval $t_1 < t < t_2$ is $(\mathrm{SRTT} \cdot S_0/2)$. Consequently, $wdsn > 0$ and the transmission rate is not increased.
- $t_2 \leq t < t_3$ (where $t_3 = t_1 + \mathrm{SRTT}$)
  $(S_0/2 \leq S \leq S_0, wdsn = 0, \mathrm{state} = \mathrm{Steady})$.
  In this time interval, the source receives the ACKs for the other $(\mathrm{SRTT} \cdot S_0/2)$ dummy packets transmitted during the Detected phase. Since $wdsn$ value is 0, the source can increase its transmission rate by $1/\mathrm{SRTT}$ each time it receives an ACK for a dummy packet.
- $t = t_3$
  $(S = S_0, wdsn = 0, \mathrm{state} = \mathrm{Steady})$.
  At time $t_3$, the source has received the ACKs for all the $(n_{\mathrm{Dummy}} = \mathrm{SRTT} \cdot S_0)$ dummy packets transmitted in the time interval $[t_0, t_1]$. Accordingly, the transmission rate has been increased by $(\Delta S = S_0/2)$; in fact

$$\begin{aligned}\Delta S &= (n_{\mathrm{Dummy}} - wdsn_0) \cdot \frac{1}{\mathrm{SRTT}} \\ &= (\mathrm{SRTT} \cdot S_0 - \mathrm{SRTT} \cdot S_0/2) \cdot \frac{1}{\mathrm{SRTT}} \\ &= \frac{S_0}{2}. \end{aligned} \tag{7}$$

  Consequently, with the help of dummy packet transmission during the Detected state, the RCS source resumes its data transmission rate value it had before the data packet loss was detected.

To observe this behavior, we perform simulation experiments whose network architecture details and parameters are explained in detail in Section V. The experiments are performed with assuming $S_0 = 22$ packets/s and $\mathrm{RTT} = 550$ ms. In Fig. 12, we show the transmission rate, $S$, dependent on time when a packet loss is due to link errors. As explained in Sections III-C and III-D, the data rate halved with a detection of a packet loss is recovered by the help of dummy packets transmitted in the Detected state. Hence, we observe this behavior in Fig. 12 that the RCS source returns to its previous rate within approximately two RTTs when a packet loss occurs due to link errors.

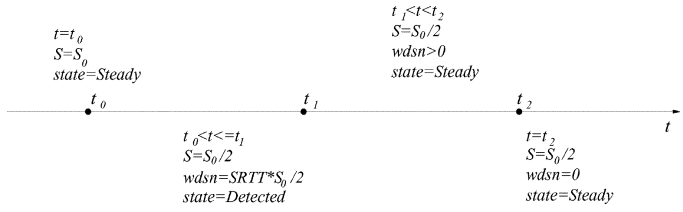


Fig. 13. RCS behavior when a packet loss occurs due to network congestion.

### *B. Packet Loss Due to Network Congestion*

Here we describe the behavior of the RCS algorithms when a packet loss occurs due to network congestion. We show that RCS source, indeed, halves its transmission rate and hence maintains the TCP-friendly behavior when the network is congested. The detailed trace of RCS protocol operation in this scenario is presented as follows.

Consider a single connection and let $t$ be time instance given in Fig. 13.

- $t = t_0$
  $(S = S_0, \mathrm{state} = \mathrm{Steady})$.
  Let $S_0$ denote the data transmission rate at time $t_0$.
- $t_0 < t \leq t_1$ (where $t_1 = t_0 + \mathrm{SRTT}$)
  $(S = S_0/2, wdsn = \mathrm{SRTT} \cdot S_0/2, \mathrm{state} = \mathrm{Detected})$.
  Suppose that at time $t_0$ the source detects a packet loss due to network congestion, i.e., the connection path can accommodate at most a transmission rate given by $S_0$. The source enters the Detected state, halves its transmission rate, i.e., $S = S_0/2$, and sets $wdsn$ to $wdsn_0 = \mathrm{SRTT} \cdot S_0/2$. Moreover, it transmits $(\mathrm{SRTT} \cdot S_0)$ dummy packets at a rate equal to $S_0$. Consequently, the overall transmission rate is $(3 \cdot S_0/2)$. However, the connection path can accommodate at most a rate given by $S_0$. Since data packets have high priority they are not discarded, whereas the half of the dummy packets (because they have low priority) will be discarded.
- $t_1 < t < t_2$ (where $t_2 = t_1 + \mathrm{SRTT}$)
  $(S = S_0/2, wdsn > 0, \mathrm{state} = \mathrm{Steady})$.
  At time $t = t_1$, the RCS source returns to the Steady state and starts to receive the ACKs for the $(\mathrm{SRTT} \cdot S_0/2)$ dummy packets transmitted in the time interval $[t_0, t_1]$ which are not discarded by the congested router. Since $wdsn > 0$ in the time interval $[t_1, t_2]$, then the transmission rate will not be increased.
- $t \geq t_2$
  $(S = S_0/2, wdsn = 0, \mathrm{state} = \mathrm{Steady})$.
  By the time $t_2$, all the ACKs for dummy packets which are not dropped by the network reach the source. The value of $wdsn$ has always been higher than zero when the ACKs for dummy packets were received. Consequently, the transmission of the dummy packets during the detected state does not cause any increase in the transmission rate hence RCS preserves TCP-friendliness in case of network congestion.

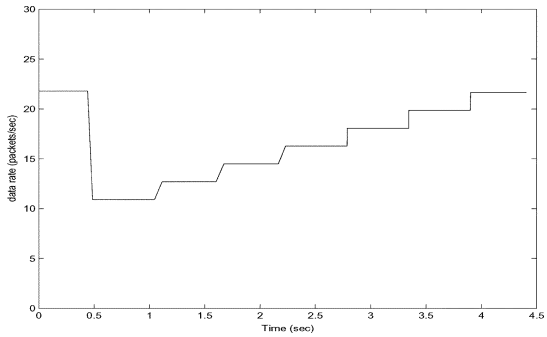


Fig. 14. Transmission rate when a packet loss is due to network congestion.

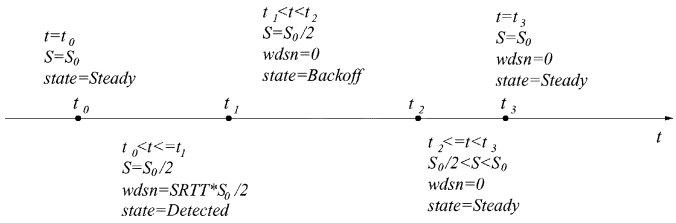


Fig. 15. RCS behavior when a packet loss occurs due to temporary signal loss.

In Fig. 14, we show the data transmission rate, $S$, dependent on time when a data packet loss occurs due to network congestion. Fig. 14 is obtained through simulation assuming $S_0 = 22$ packets/s and $\mathrm{RTT} = 550$ ms. At time $t_0 = 0.5$ s, a packet loss is detected, accordingly, the transmission rate, $S$, is set to $S = 11$ packets/s. As explained in Section III-D, by the help of protocol variable $wdsn$, the RCS source does not increase the rate for the first half of the dummy ACKs it receives during the Steady state. Since the congestion is experienced, the second half of the dummy packets transmitted are discarded and hence no rate increase is performed with the reception of dummy ACKs during the Steady state. Therefore, for $t > t_0$, the transmission rate increases by $\{\text{one packet}\}/\mathrm{RTT}$ each RTT as in the TCP case, i.e., RCS behavior is TCP-friendly.

### C. Packet Loss Due to Temporary Signal Loss

Here we assume that the packet loss occurs due to temporary signal loss and describe the resulting behavior of the proposed protocol.

Let $t$ be time instance in Fig. 15.

- $t = t_0$
  $(S = S_0, \mathrm{state} = \mathrm{Steady})$.
  Suppose that the data transmission rate is $S_0$ at time $t_0$.
- $t_0 < t \leq t_1$ (where $t_1 = t_0 + \mathrm{SRTT}$)
  $(S = S_0/2, wdsn = \mathrm{SRTT} \cdot S_0/2, \mathrm{state} = \mathrm{Detected})$.
  Suppose at time $t_0$ the source detects a packet loss due to temporary signal loss. The source enters the Detected state, halves its transmission rate, i.e., $S = S_0/2$, and sets $wdsn$ to $(wdsn_0 = \mathrm{SRTT} \cdot S_0/2)$.
- $t_1 < t < t_2$ (where $t_2$ is the time when the first ACK is received).
  $(S = S_0/2, wdsn = 0, \mathrm{state} = \mathrm{Backoff})$.
  Due to the signal loss, source does not receive any ACK during this period and goes to Backoff state at $t = t_1$, i.e., $\mathrm{state} = \mathrm{Backoff}$. Source transmits data and dummy packets with rate $S = S_0/2$. Once an ACK is received, source terminates Backoff phase. Since the resolution of signal loss is detected with ACK reception, $\mathrm{SRTT} \cdot S_0/2$ dummy packets are transmitted before Backoff phase is terminated.
- $t_2 \leq t < t_3$ (where $t_3 = t_2 + \mathrm{SRTT}$)
  $(S_0/2 \leq S \leq S_0, wdsn = 0, \mathrm{state} = \mathrm{Steady})$.
  During this interval, the source receives the ACKs for the dummy packets transmitted during the Backoff phase after the temporal signal loss is resolved. Since $wdsn$ value is 0, the source can increase its transmission rate by $1/\mathrm{SRTT}$ each time it receives an ACK for a dummy packet.
- $t = t_3$
  $(S = S_0, wdsn = 0, \mathrm{state} = \mathrm{Steady})$.
  At time $t_3$, the source has received the ACKs for all the $(\mathrm{SRTT} \cdot S_0/2)$ dummy packets transmitted in the time interval $[t_1, t_2]$. Hence, the original transmission rate $S_0$ is resumed in approximately SRTT period after the temporal signal loss is resolved.

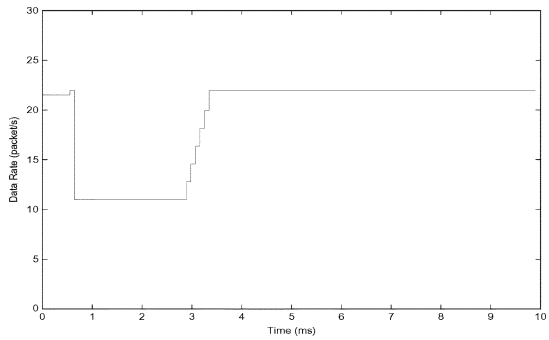


Fig. 16. Transmission rate when a packet loss is due to temporal signal loss.

To illustrate the enhanced behavior of RCS in temporal signal loss conditions, we show the transmission rate change, $S$, with time in Fig. 16. Fig. 16 is obtained by simulation assuming $S_0 = 22$ packets/s and $\mathrm{RTT} = 550$ ms. The duration of signal loss $D_{\mathrm{SignalLoss}}$ is assumed to be $2 \cdot \mathrm{SRTT}$. In this case, RCS source transmits dummy and data packets in the Backoff state with the same rate for the reasons explained in Section III-E. Hence, RCS source avoids unnecessary consecutive rate decrease in case of signal loss and resumes its original transmission rate $S_0$ within approximately one SRTT after the signal loss is resolved.

## V. Performance Evaluation

In order to investigate the performance of the RCS, we conducted extensive simulation experiments using ns-2[1] The details of the simulation environment are presented in Section V-A. Then, RCS throughput, dummy traffic overhead, fairness, and the performance achieved in case of temporal link loss conditions are evaluated in Sections V-B, V-C, V-D, and V-E, respectively.

### A. Simulation Environment

We simulate a network topology where $N$ sources transmit data to $N$ destinations over lossy and high propagation delay Geostationary (GEO) satellite links. The $N$ streams are multiplexed in the router $A$, whose buffer accommodates $K$ packets. The GEO satellite is assumed to be a *bent-pipe satellite*, i.e., it

[1]The RCS ns-2 agent can be downloaded from the following URL: http://www.ece.gatech.edu/research/labs/bwn.

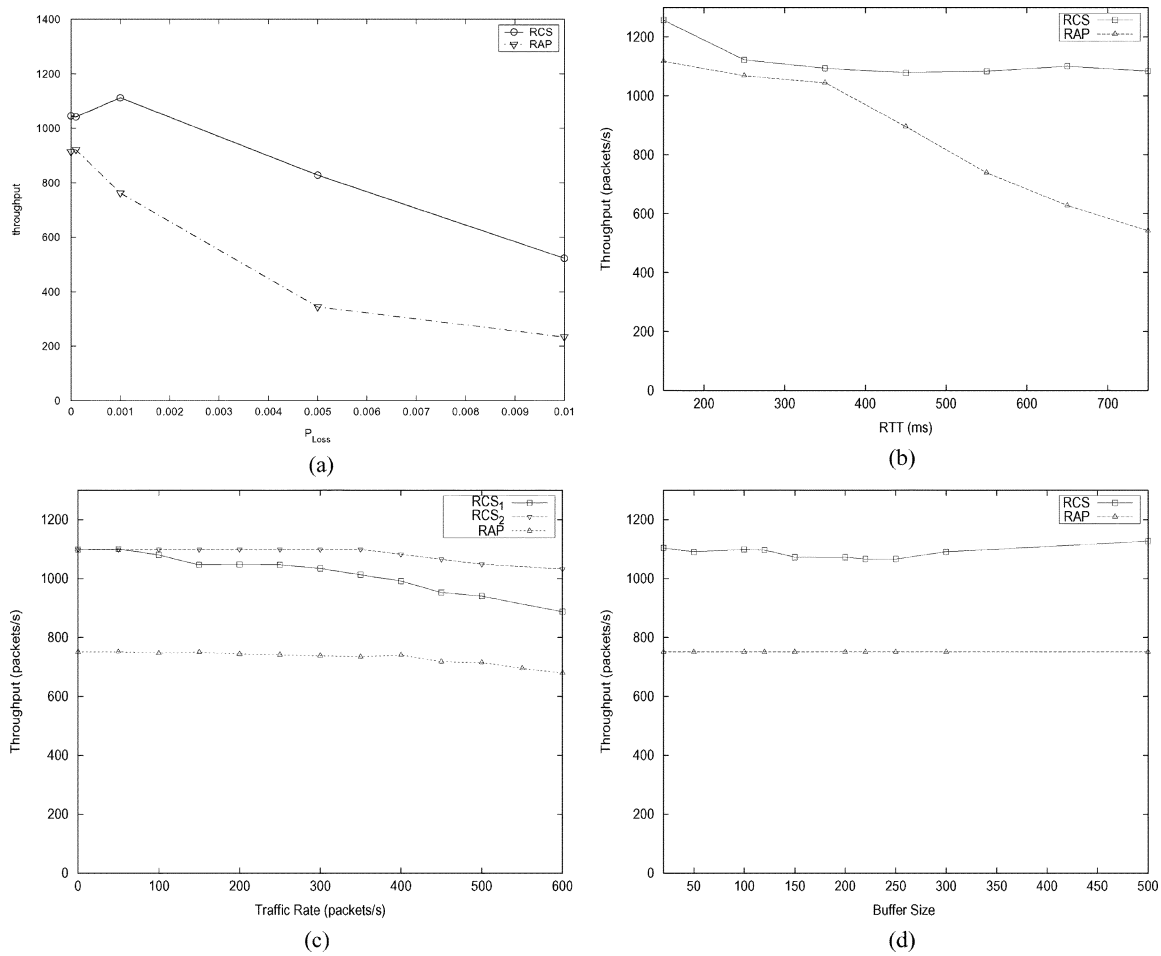


Fig. 17. Throughput Performance of RCS (solid lines) and RAP (dashed lines) for (a) varying loss probability $P_{\text{Loss}}$, (b) varying round-trip time RTT, (c) varying background low-priority traffic rate $S_L$, (d) varying buffer size $K$.

directly relays the packets without any buffering and processing. The multiplexed streams are de-multiplexed in the router $B$ and then routed to corresponding $N$ destinations. The routers are equipped with a simple priority-queuing mechanism with two priority levels, i.e., low and high priority, following first-in first-out (FIFO) queuing with priority-based preemption. In this priority-queuing mechanism, low priority packets, i.e., dummy packets, are discarded first in case of congestion when the queue is overutilized and overflown. Both data and dummy packets may get lost due to link errors with probability $P_{\text{Loss}}$ in the lossy link. We assume that $N = 10, K = 50$ packets and the link capacity is $c = 1300$ packets/s, which is approximately equal to 10 Mb/s for packets of length 1000 bytes. As an example of a long delay link, unless otherwise stated, we assume $\text{RTT} = 550$ ms, which is a typical RTT value in the GEO satellite links [1].

### *B. Throughput Performance*

Throughput performance is evaluated for different values of the packet loss probability $P_{\text{Loss}}$, the round trip time, RTT, the rate of the background low-priority traffic $S_L$, and the buffer size $K$. The results obtained are shown in Fig. 17. For the sake of comparison, we also show the performance achieved by the *rate adaptation protocol* (RAP) [20]. We choose the RAP protocol because, to the best of our knowledge, it achieves the best performance in terms of real-time multimedia delivery in the Internet.

In Fig. 17(a), we show RCS throughput for different values of loss probabilities, $P_{\text{Loss}}$[2]. As shown in Fig. 17(a), RCS significantly improves the throughput performance over RAP for all values of $P_{\text{Loss}}$. RCS throughput is maximum when $P_{\text{Loss}} = 10^{-3}$. With increasing $P_{\text{Loss}}$, the throughput of both RCS and RAP decreases since they both follow the conservative rate decrease behavior to preserve TCP-friendliness. For $P_{\text{Loss}} = 5 \cdot 10^{-3}$, RCS achieves more than 150% throughput improvement over RCS. This is mainly because the RCS algorithms efficiently distinguishes the cause of the packet loss and recovers from the unnecessary rate throttles by the help of dummy packets.

In Fig. 17(b), we show RCS throughput for varying RTT values, i.e., $100 \text{ ms} \leq \text{RTT} \leq 750$ ms. Here, we assume $P_{\text{Loss}} = 10^{-3}$. RCS throughput is not significantly affected by increasing RTT while RAP throughput severely degrades for high RTT values, i.e., for $\text{RTT} \geq 250$ ms. Such serious performance loss experienced by RAP is due to the amplifying effects of the high propagation delay on the performance degradation due to link errors. On the contrary, as explained in Section IV-A, RCS can quickly recover from the rate halving it performs in case of packet loss due to link errors by the help of dummy packets. Furthermore, note that RCS also outper-

[2]The bit error rate (BER) in satellite networks can be as high as $10^{-4}$, i.e., one bad bit out of 10 000 bits. For packets of 1000 bytes, the BER $10^{-4}$ gives a packet loss probability, $P_{\text{Loss}}$ higher than $10^{-2}$ even if powerful FEC algorithm is applied.

forms RAP for moderate RTT values such as $\mathrm{RTT} = 150$ ms as shown in Fig. 17(b). Therefore, while the algorithms of RCS are specifically tailored to address the challenges in networks with high bit error rates and high bandwidth-delay products, high throughput performance can still be achieved in case of moderate RTT values.

In order to investigate the effects of the low-priority background traffic on the RCS performance, we now assume that low-priority background traffic, which is generated by an application at the data rate of $S_L$ packets/s, flows through the satellite channel. We consider two scenarios. In the first one, the low-priority background traffic sources are generating UDP traffic at a fixed data rate $S_L$ without performing any congestion control. In the second scenario, we assume the background low-priority traffic sources are TCP-friendly and perform congestion control. Here, we assume $P_{\mathrm{Loss}} = 10^{-3}$ and $\mathrm{RTT} = 550$ ms.

For the first scenario, we consider a wide range of $S_L$ values, i.e., $0 \leq S_L \leq 600$ packets/s. As shown in Fig. 17(c), the throughput performance of RCS ($\mathrm{RCS}_1$ curve) is not significantly affected by the low-priority traffic for moderate background traffic rates, i.e., $S_L = 400$ packets/s. For $S_L > 400$ packets/s, RCS throughput degrades slightly. This is mainly because the low-priority background traffic in this scenario does not perform congestion control and hence creates congestion in the network. Because of very high low-priority traffic rate, higher number of dummy packets, which are transmitted by the RCS for link probing, are also dropped at the routers. However, as it is observed in Fig. 17(c), this degradation is very slight even for $S_L = 600$ packets/s and RCS always outperforms RAP for all values of $S_L$.

In the second scenario, we perform simulation experiments using the TCP-friendly sources generating low-priority background traffic in the scenario given in Section V-A. We use RAP as a rate control protocol at the background traffic sources to generate responsive low-priority traffic. As shown in Fig. 17(c), the throughput of RCS ($\mathrm{RCS}_2$ curve) is not affected by the background traffic and remains almost constant. This is because the low-priority traffic also performs rate control in case of congestion and hence it does not significantly affect the dummy traffic.

Furthermore, we investigate the throughput performance for varying buffer size $K$. We assume $P_{\mathrm{Loss}} = 10^{-3}, \mathrm{RTT} = 550$ ms, $20 \leq K \leq 500$. As shown in Fig. 17(d), RCS throughput does not change significantly for $K \leq 300$. However, for higher buffer sizes, RCS throughput slightly increases. This is mainly because of the rate-based nature of the RCS. Since RCS can recover from the rate throttle due to link errors, it can more efficiently utilize link resources in the links with high bit error rates and high propagation delays. Thus, an increase in the buffer size can further contribute to this link utilization efficiency. Note also that RCS outperforms RAP for all values of buffer sizes.

### C. Dummy Packet Traffic

Let $N_{\mathrm{Dummy}}$ and $N_{\mathrm{Data}}$ be the total number of transmitted dummy and data packets, respectively. Then, the overhead is defined as $\mathrm{overhead} = (N_{\mathrm{Dummy}})/(N_{\mathrm{Data}} + N_{\mathrm{Dummy}})$.

In the upper plot of Fig. 18, we show the overhead dependent on the loss probability, $P_{\mathrm{Loss}}$. Obviously, the overhead increases when $P_{\mathrm{Loss}}$ increases. This is mainly because as $P_{\mathrm{Loss}}$ increases higher number of packet losses are detected and hence RCS sources enter Detected state more frequently. This leads to

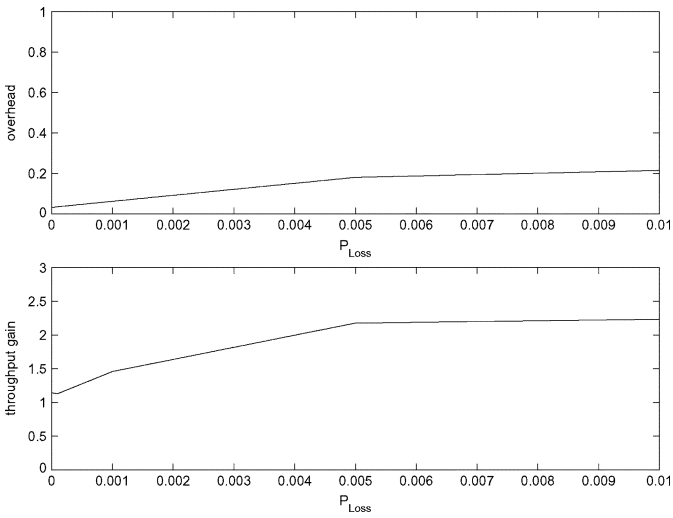


Fig. 18. Comparison of bandwidth overhead and throughput gain between RCS and RAP.

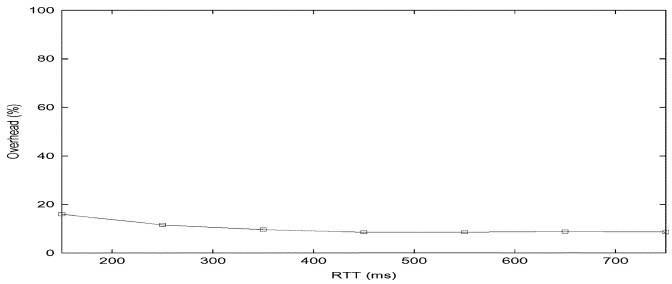


Fig. 19. Overhead due to dummy packet traffic for varying RTT.

transmission of higher number of low priority dummy packets as explained in Section III-D. Note that the overhead can be as high as 21.5% when $P_{\mathrm{Loss}} = 10^{-2}$. However, using dummy packets RCS achieves much higher throughput than other rate control schemes for real-time traffic. For example, in the bottom plot of Fig. 18, we show the throughput gain obtained using RCS. We evaluated the throughput gain as the ratio between the throughput achieved by RCS and the throughput achieved by RAP. Note that when $P_{\mathrm{Loss}} = 10^{-2}$, the overhead is 21.5%, but the throughput gain is higher than 200%.

We also perform simulation experiments to investigate the effects of RTT on the amount of created dummy packet traffic. As shown in Fig. 19, the overhead created by the dummy traffic is higher for low RTT values, i.e., $\mathrm{RTT} \approx 100$ ms. As RTT increases, the overhead drops to as low as 10% for $\mathrm{RTT} \geq 400$ ms. This is because the frequency of entering Detected state and hence the number of dummy packets transmitted decrease with increasing RTT. This result is also consistent with the fact that RCS is mainly developed for the links with high bit error rates and high propagation delay links.

### D. Fairness

For the fairness performance of RCS protocol, we consider two different scenarios, i.e., homogeneous and heterogeneous fairness.

*1) Homogeneous Scenario:* In this scenario, all connections pass through the same path and run RCS protocol in the simulation environment described in Section V-A. We

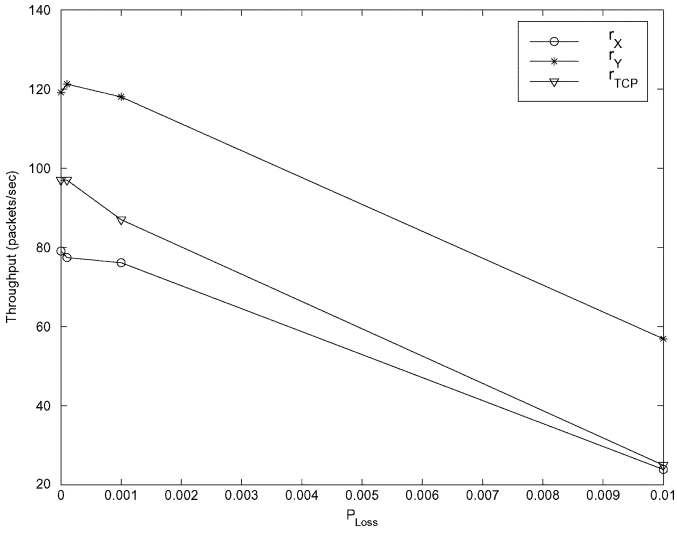


Fig. 20. Fairness between RCS and traditional TCP connections.

observe that at any time, $t$, all connections have been acknowledged the same number of data packets approximately, i.e., $\text{acked}_{i'}(t) \approx \text{acked}_{i''}(t)$, for any connection $i'$ and $i''$ [25]. This means that each RCS connection is given a fair share of network resources. Here, we assumed $N = 10, K = 50$ packets, $c = 1300$ packets/s, $P_{\text{Loss}} = 0$ and $\text{RTT} = 550$ ms. Note that we obtained similar results for several other cases.

*2) Heterogeneous Scenario:* To investigate the fairness of RCS in heterogeneous scenarios, we consider the simulation scenario in which there are $M$ connections of type X and $N$ connections of type Y. Connections of type $j$, for $j = X, Y$, are characterized by round-trip time equal to $\text{RTT}_j$, and loss probability for link errors equal to $P_{\text{Loss},j}$. All connections pass through the link which is assumed to be the bottleneck and whose capacity is assumed to be $c = 1300$ packets/s. The fairness, $\phi$, is the ratio between the average throughput of connections of type X, $r_X$, and the average throughput of connections of type Y, $r_Y$, i.e., $\phi = (r_X/r_Y)$. It is obvious that the fairness becomes higher as $\phi$ approaches 1.

In the first experiment scenario, we assume that both connections X and Y pass through the same type of lossy and high propagation delay link, i.e., $\text{RTT} = 550$ ms, but connections of type X are TCP connections and connections of type Y are RCS connections. Moreover, we assume $N = M = 5$. In Fig. 20, we see that $r_Y$ values are higher than $r_X$ thus, resources are not shared equally between TCP and RCS connections. This is mostly due to the problems of TCP in networks with high bandwidth-delay product and high link error rate. In fact, in Fig. 20, we show that $r_X \approx r_{\text{TCP}}$, where $r_{\text{TCP}}$ is the average throughput when all connections, i.e., both X and Y connections, use TCP. This means that RCS significantly improves network efficiency without penalizing TCP flows and the degradation of the TCP share is mainly because of its shortcomings when applied to the networks with high bit error rates and high propagation delays as discussed in Section II.

Hence, in order to further investigate the fairness of the RCS to TCP connections, we perform simulation experiments using the same simulation environment with different configurations. In this scenario, we assume that connections X, i.e., RCS sources, are connected to router A via

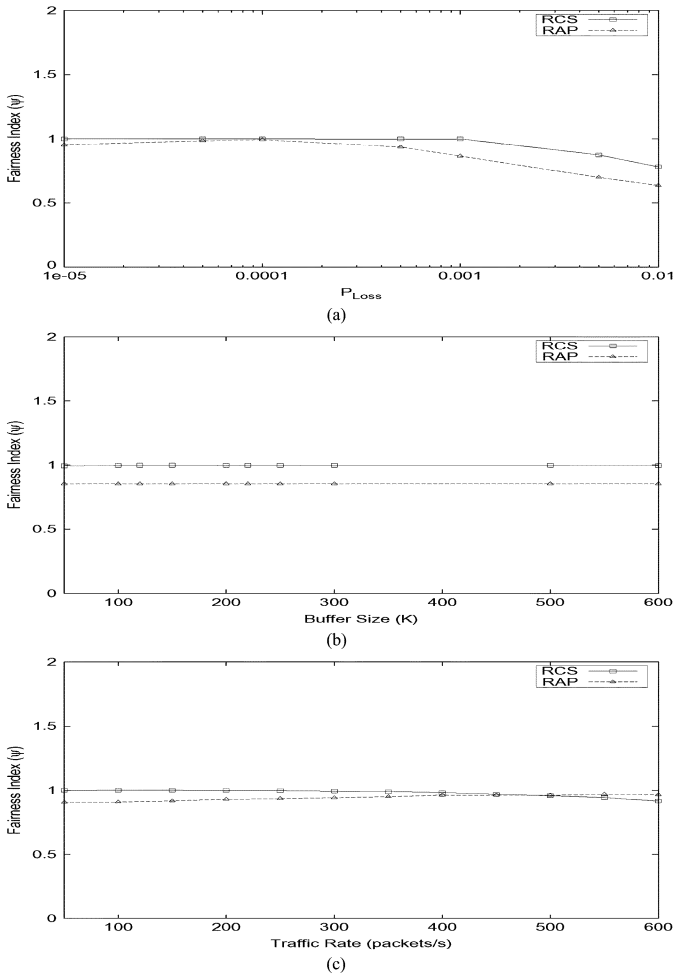


Fig. 21. Fairness Index $\psi$ between RCS and TCP connections, where TCP sources do not experience the same link conditions with high bit error rates and high propagation delays, for (a) varying packet loss probability $P_{\text{Loss}}$ (b) varying buffer size $K$ (c) varying background low-priority traffic rate $S_L$.

geostationary satellite links and connections Y, i.e., TCP sources, are connected to router A via error-free wired links. Thus, we assume $\text{RTT}_X = 550$ ms, $\text{RTT}_Y = 110$ ms, $P_{\text{Loss},X} = 10^{-3}, P_{\text{Loss},X} = 0$, and $N = M = 5$. The objective of such simulation configuration is to assess the fairness between RCS and TCP sources which are not experiencing link errors or high propagation delays.

To assess the fairness performance of RCS, we use *Jain's fairness index* [9] which quantifies the degree of similarity between the amount of link resources used by all connections. Let $x_i$ be the throughput of the $i$th connection and let $n$ be the number of connections competing for the same bottleneck resources. Then, the *Jain's Fairness Index*, $\psi$, can be evaluated as follows:

$$\psi = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot \sum_{i=1}^{n} x_i^2} \tag{8}$$

When the fairness index is 1, all of the connections consume the same amount of bottleneck resources. The fairness index $\psi$ decreases as the difference between the throughput values achieved by different connections increases.

First, we explore the fairness index of RCS versus $P_{\text{Loss}_X}$. As it is observed in Fig. 21(a), RCS fairly shares the link resources with TCP sources for $P_{\text{Loss}_X} < 0.001$, i.e., $\psi \approx 1$. For $P_{\text{Loss}_X} \geq 0.001$, the fairness index decreases as the throughput

performance of the RCS degrades with increasing $P_{\mathrm{Loss}_X}$ as also observed in Section V-B. The same experiments are also performed with RAP sources instead of RCS. Note that the fairness index obtained with RAP sources is lower than RCS for $P_{\mathrm{Loss}_X} \geq 0.001$ as shown in Fig. 21(a). This is again because of the fact that RAP cannot adapt itself to high bit error rate conditions and hence experience throughput degradation as observed in Section V-B.

Under the same simulation scenario, we also investigate the effects of buffer size $K$ on the fairness of the RCS to the TCP sources. As shown in Fig. 21(b), all of the connections consume almost the same amount of bottleneck resources for all buffer sizes, i.e., $\psi \approx 1$ for $K \leq 600$. This is mainly because the RCS throughput is not significantly affected by the buffer size as also observed in Fig. 17(d). Note that the same behavior is also observed for the experiments performed with RAP sources as shown in Fig. 21(b). However, note also that the fairness index is lower than RCS case, which is due to the degraded throughput performance of RAP protocol in the links with high propagation delays and high bit error rates.

Furthermore, we investigate the effects of background low-priority traffic on the fairness performance of the RCS. In this scenario, the UDP sources create background low-priority traffic with constant data rate $S_L$ as in Section V-B. We perform the simulation experiments for $0 \leq S_L \leq 600$ packets/s. As shown in Fig. 21(c), RCS link share does not exceed that of TCP for $S_L > 0$. Furthermore, RCS link share decreases with increasing $S_L$ and hence the fairness index $\psi$ slightly decreases as shown in Fig. 21(c). This is because the RCS throughput slightly decreases with increasing low-priority background traffic rate as it was also observed in Section V-B.

### E. Temporal Link Blockage

Here, we simulate the same network topology given in Section V-A with varying signal loss durations $D_{\mathrm{SignalLoss}}$. Here, we assume $P_{\mathrm{Loss}} = 10^{-3}$ and $\mathrm{RTT} = 550$ ms. In Fig. 22(a), we show the throughput of RCS and RAP [20] for signal loss of duration $0 \leq D_{\mathrm{SignalLoss}} \leq 60$ seconds. Note that while RAP performance throughput decreases dramatically as the signal loss duration increases, the performance degradation of RCS is negligible.

In order to assess the advantages provided by the Backoff algorithm, in Fig. 22(a) we also show the performance of RCS if the Backoff algorithm is not implemented. The Backoff algorithm gives significant performance increase because it avoids additional transmission rate throttles until an ACK is received indicating that the signal has been recovered. At $D_{\mathrm{SignalLoss}} = 30$ seconds, RCS outperforms RAP by more than 260% throughput improvement.

We also evaluated the overhead caused by the dummy packet traffic for varying $D_{\mathrm{SignalLoss}}$. As shown in Fig. 22(b), the overhead due to low-priority dummy packets slightly increases with increasing duration of temporal link loss. This is because RCS stays in the Backoff state and continuously transmits dummy packets until the link is back as explained in Section III-E. However, by means of the additional traffic caused by the dummy packets transmitted during Backoff state, the throughput gain achieved in case of temporal signal loss situations is very significant as shown in Fig. 22(a).

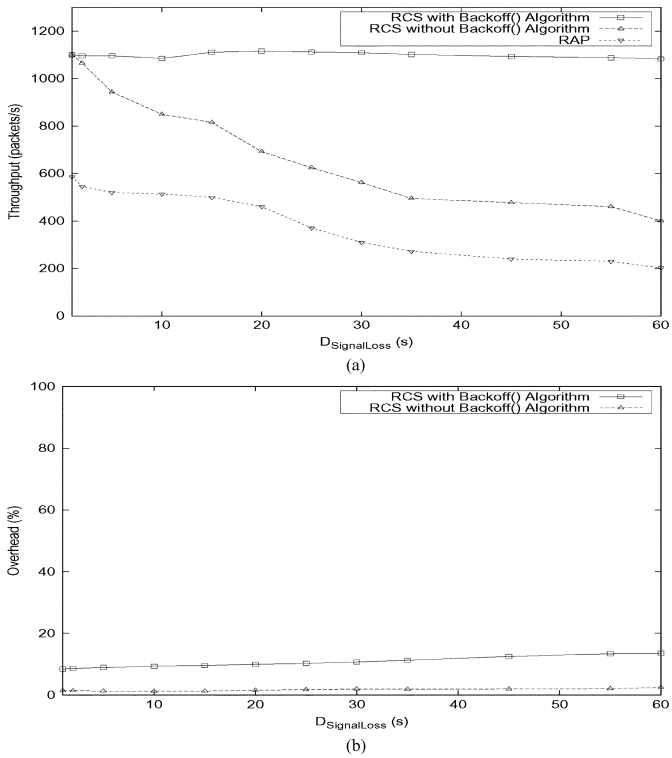


Fig. 22. (a) Throughput performance of RCS with and without `Backoff()` algorithm and RAP for different values of signal loss duration, $D_{\mathrm{SignalLoss}}$. (b) The amount of dummy packet traffic in temporal link loss conditions for different values of $D_{\mathrm{SignalLoss}}$.

## VI. Conclusion

In this paper we introduced RCS, a new rate control scheme for real-time traffic in networks with high bandwidth-delay products and high link error rates. RCS improves throughput using low priority dummy packets to probe available network resources. Therefore, RCS requires the routers along the connection to implement some priority mechanism.[3] The main feature of RCS is that it is an end-to-end protocol, i.e., it needs to be implemented only at the source and destination. RCS is a TCP-friendly rate control scheme, which halves its transmission rate in case of packet loss. RCS can then resume its original rate very rapidly in case of packet losses due to link errors. If the packet loss is due to congestion, RCS follows TCP-friendly behavior rules and increases transmission rate additively after rate halving. For temporal signal loss situations, RCS avoids unnecessary rate throttles and resumes its original transmission rate very rapidly after signal is back. Simulation results show that RCS achieves high performance in terms of throughput and fairness for real-time multimedia applications in networks with high bandwidth-delay product and high link error rate while providing TCP-friendly behavior.

## References

[1] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-peach: A new congestion control scheme for satellite IP networks," *IEEE/ACM Trans. Networking*, vol. 9, no. 3, pp. 307–321, Jun. 2001.

[3]While it is immediate that RCS interacts properly with active queue management (AQM) schemes considering different levels of priority, e.g., RIO [10], its interactions with other AQM solutions are under investigation.

[2] I. F. Akyildiz, X. Zhang, and J. Fang, "TCP-Peach+: Enhancement of TCP-Peach for satellite IP networks," *IEEE Commun. Lett.*, vol. 6, no. 7, pp. 303–305, Jul. 2002.
[3] A. Bakre and B. R. Badrinath, "I-TCP: Indirect-TCP for mobile hosts," in *Proc. 15th Int. Conf. Distributed Computing Systems*, May 1995, pp. 136–143.
[4] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for internet hosts," in *Proc. ACM SIGCOMM*, Sep. 1999, pp. 175–187.
[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 756–769, Dec. 1997.
[6] G. Bianchi, A. Capone, and C. Petrioli, "Throughput analysis of end-to-end measurement-based admission control in IP," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 2000, pp. 1461–1470.
[7] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint admission control: Architecture issues and performance," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, 2000, pp. 57–70.
[8] S. Cen, C. Pu, and J. Walpole, "Flow and congestion control for internet media streaming applications," in *Proc. SPIE Multimedia Computing and Networking*, Jan. 1998, pp. 250–264.
[9] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithm for congestion avoidance in computer networks," *Comput. Netw. ISDN Syst. J.*, vol. 17, no. 1, pp. 1–14, Jun. 1989.
[10] D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 362–373, Aug. 1998.
[11] T. Ferrari, W. Almesberger, and J. Y. Le Boudec, "SRP: A scalable resource reservation protocol for the internet," *Comput. Commun.*, vol. 21, no. 14, pp. 1200–1211, Sep. 1998.
[12] S. Floyd, "TCP and explicit congestion notification," *ACM Comput. Commun. Rev.*, vol. 24, no. 5, pp. 10–23, Oct. 1994.
[13] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM*, Aug. 2000, pp. 45–58.
[14] S. Jacobs and A. Eleftheriadis, "Real-time dynamic rate shaping and control for internet video applications," in *Proc. Workshop on Multimedia Signal Processing*, Jun. 1997, pp. 23–25.
[15] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov, "TCP over second (2.5G) and third (3G) generation wireless networks," IETF, Dec. 2002. Internet-Draft draft-ietf-pilc-2.5g3g-12.
[16] H. Kanakia, P. Mishra, and A. Reibman, "An adaptive congestion control scheme for real-time packet video transport," in *Proc. ACM SIGCOMM*, San Francisco, CA, Sep. 1993, pp. 20–31.
[17] Cooperative Association for Internet Data Analysis Path and Round-Trip Time Measurements (2004, Apr.). [Online]. Available: http://www.caida.org/outreach/presentations/IEPG.9808/
[18] J. Padhye, V. Firoio, D. Towsley, and J. Kurose, "Modeling TCP reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Networking*, vol. 8, no. 2, pp. 133–145, Apr. 2000.
[19] R. Puri, K.-W. Lee, K. Ramchandran, and V. Bharghavan, "An integrated source transcoding and congestion control paradigm for video streaming in the internet," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 18–32, Mar. 2001.
[20] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 1999, pp. 1337–1345.
[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport for real-time applications," RFC 1889, Jan. 1996.
[22] P. Sinha, N. Venkataraman, R. Sivakumar, and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," in *Proc. ACM MOBICOM*, Seattle, WA, Aug. 1999, pp. 231–241.
[23] D. Sisalem and H. Schulzrinne, "The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme," presented at the Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Cambridge, U.K., Jul. 1998.
[24] P. A. Spagnolo, T. R. Henderson, J. H. Kim, and G. T. Michael, "TCP gateway design considerations for satellite link blockage mitigation," in *Proc. IEEE Int. Conf. Communications*, vol. 1, May 2003, pp. 433–437.
[25] J. Tang, G. Morabito, I. F. Akyildiz, and M. Johnson, "RCS: A rate control scheme for real-time traffic in networks with high bandwidth-delay products and high bit error rates," in *Proc. IEEE INFOCOM*, vol. 1, Apr. 2001, pp. 114–122.

**Ian F. Akyildiz** (M'86–SM'89–F'96) received the B.S., M.S., and Ph.D. degrees in computer engineering from the University of Erlangen-Nuernberg, Germany, in 1978, 1981, and 1984, respectively.

Currently, he is the Ken Byers Distinguished Chair Professor of the School of Electrical and Computer Engineering, and Director of the Broadband and Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta. His current research interests are in sensor networks, InterPlaNetary Internet, wireless networks, satellite networks and the next-generation Internet. He is the Editor-in-Chief of *Computer Networks* (Elsevier), and the founding Editor-in-Chief of *Ad Hoc Networks Journal* (Elsevier).

Dr. Akyildiz has been a Fellow of the ACM since 1996. He was the technical program chair and general chair for several IEEE and ACM conferences including IEEE INFOCOM, ACM MOBICOM, and IEEE ICC. He received the Don Federico Santa Maria Medal for his services to the Universidad of Federico Santa Maria in Chile in 1986. He served as a National Lecturer for ACM from 1989 until 1998 and received the ACM Outstanding Distinguished Lecturer Award for 1994. He received the 1997 IEEE Leonard G. Abraham Prize award (IEEE Communications Society) for his paper entitled "Multimedia Group Synchronization Protocols for Integrated Services Architectures" published in the IEEE JOURNAL OF SELECTED AREAS IN COMMUNICATIONS (JSAC) in January 1996. He received the 2002 IEEE Harry M. Goode Memorial award (IEEE Computer Society) with the citation "for significant and pioneering contributions to advanced architectures and protocols for wireless and satellite networking." He received the 2003 IEEE Best Tutorial Award (IEEE Communications Society) for his paper entitled "A survey on sensor networks," published in *IEEE Communications Magazine*, in August 2002. He also received the 2003 ACM Sigmobile Outstanding Contribution Award with the citation "for pioneering contributions in the area of mobility and resource management for wireless communication networks."

**Özgür B. Akan** (M'00) received the B.Sc. and M.Sc. degrees in electrical and electronics engineering from Bilkent University and Middle East Technical University, Ankara, Turkey, in 1999 and 2001, respectively. He received the Ph.D. degree in electrical and computer engineering from the Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, in 2004.

He is currently an Assistant Professor with the Department of Electrical and Electronics Engineering, Middle East Technical University. His current research interests include next-generation wireless networks, wireless sensor networks, and deep space communication networks.

**Giacomo Morabito** received the Laurea degree in electrical engineering and the Ph.D. degree in electrical, computer, and telecommunications engineering from the University of Catania, Italy, in 1996 and 2000, respectively.

From November 1999 to April 2001, he was with the Broadband and Wireless Networking Laboratory of the Georgia Institute of Technology as Research Engineer. Since May 2001, he has been with the School of Engineering at Enna of the Unversity of Catania, where he is currently an Assistant Professor. His research interests include mobile and satellite networks, self-organizing networks, quality of service and traffic management.

Dr. Morabito is serving on the editorial boards of *Computer Networks* and *IEEE Wireless Communications Magazine*, as a guest editor of *Computer Networks* and MONET, and as a member of the technical program committee of several conferences. He was the Technical Program Co-Chair of Med-Hoc-Net 2004.