ARC: The Analytical Rate Control Scheme for Real-Time Traffic in Wireless Networks

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

***Abstract*—Next-generation wireless Internet (NGWI) is expected to provide a wide range of services including real-time multimedia to mobile users. However, the real-time multimedia traffic transport requires rate control deployment to protect shared Internet from unfairness and further congestion collapse. The transmission rate control method must also achieve high throughput and satisfy multimedia requirements such as delay or jitter bound. However, the existing solutions are mostly for the wired Internet, and hence, they do not address the challenges in the wireless environments which are characterized by high bit error rates. In this paper, a new analytical rate control (ARC) protocol for real-time multimedia traffic over wireless networks is presented. It is intended to achieve high throughput and multimedia support for real-time traffic flows while preserving fairness to the TCP sources sharing the same wired link resources. Based on the end-to-end path model, the desired behavior of a TCP source over lossy links is captured via renewal theory. The resulting asymptotic throughput equation is designated as the driving equation for the proposed rate control method. Performance evaluation via simulation experiments reveals that ARC achieves high throughput and meets multimedia traffic expectations without violating good citizenship rules for the shared Internet.**

***Index Terms*—Equation-based rate control, jitter bound, real-time multimedia, wireless networks.**

I. Introduction

HIGHER sensitivity to time constraints such as delay and jitter and more tolerance to packet losses are the characteristics of real-time multimedia traffic. These specifications also constitute the reason why there is no transmission rate control for multimedia applications. This fact would not lead to a problem, if the quality of service (QoS)-oriented and reservation-based *DiffServ* [7] or *IntServ* [33] architectures were used. However, there will always exist a necessity for a rate control mechanism for real-time flows to avoid unfair resource sharing among competing sources and congestion collapse. The transmission rate control can be in cooperation with encoding process. Hence, the real-time multimedia encoder can then adapt its encoding rate in accord with the controlled fair share of network resources [30].

Manuscript received October 29, 2002; revised May 22, 2003 and September 1, 2003; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Krunz. This work was supported by the National Science Foundation under Grant ANI-0117840.

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, 06531 Ankara, Turkey (e-mail: akan@eeemetu.edu.tr).

I. 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).

Digital Object Identifier 10.1109/TNET.2004.833155

The need of rate control for real-time flows in the Internet has been subject of many papers in the literature [5], [9], [22], [23]. However, most of the effort has been put into the rate control research for wireline networks rather than wireless networks. The common feature of these proposals is that they follow the conservative rate halving behavior of TCP. This behavior is inherited from the assumption of reliable and almost error-free communication link presence. This leads to recognition of a single packet loss as a congestion indication, which may well not hold in error-prone wireless links. Therefore, TCP-like rate control for wireless networks leads to unnecessary rate throttle, and hence, severe performance degradation. One way to overcome this problem is to distinguish packet losses due to link errors from congestion. Although there are some proposed methods to achieve this [12], [20], [25], it is not easy to obtain high accuracy in determination of actual reason for packet loss to take proper action. In [28], Rate Control Scheme (RCS) has been proposed for real-time traffic in the networks with high bandwidth-delay products and high bit error rates. RCS significantly improves the throughput performance while maintaining fairness. However, its *dummy packet* based congestion control algorithm is specifically tailored to match the requirements of the typical satellite links with high propagation delay and may be inefficient for the wireless environments with low access delay such as WLAN and pico-cells.

In addition to additive-increase multiplicative-decrease (AIMD) rate control proposals in the literature, some research has also been performed on equation-based congestion control [14], [21], [29], [34]. Instead of responding to a single packet loss, equation-based congestion control uses a control equation to adjust data rate. TCP-Friendly Rate Control (TFRC) [14] is the most important equation-based congestion control proposal in the literature. The control equation of TFRC is the TCP response function [21] governing the steady-state transmission rate of TCP as a function of round-trip time (RTT) and loss event rate p. The loss event rate is calculated at the receiver and sent to the sender. The source, having loss event rate and RTT, can adjust its data rate by using its control equation. Hence, TFRC achieves fairness and smooth rate change with equation-based congestion control.

The equation-based rate control can also be used in wireless networks to address the requirements of real-time multimedia transport and the characteristics of wireless links. However, the existing equation-based rate control schemes cannot be directly applied to the wireless environments. The fundamental reason is that the throughput equation in [21] models the steady-state TCP behavior over error-free wireline links. This approach is again based on the assumption that a packet loss is an indication of congestion and hence is directly related to the actual link re-

sources. For instance, TFRC [14] uses the loss event rate, which is calculated by considering packet loss events observed at the receiver. In a wireless scenario, this approach cannot distinguish the packet loss reasons and the calculated loss event rate contains also the packet losses due to wireless link errors. Therefore, it may result in inaccurate rate determination and hence underutilization of the link resources.

There also exist some studies in the literature that model the TCP behavior over lossy wireless channels. For instance, a stochastic analytical model of TCP-Reno over lossy links is presented in [1]. In [17], TCP performance over networks with high bandwidth-delay products and random loss is analyzed. Nevertheless, the plain TCP behavior over wireless links is not desirable since TCP itself is not well suited to lossy links. Therefore, the response function obtained from the throughput model of TCP over wireless links should not be the control equation to achieve efficient analytical rate control. The required throughput model should reflect the desired TCP behavior over lossy links, i.e.,

- TCP source should not throttle data rate in case of packet loss due to wireless link error;
- it should follow standard TCP rules otherwise.

Hence, what should be modeled is not the actual TCP behavior, but rather its desired behavior over wireless links. The throughput equation obtained from such a model can achieve throughput efficient and TCP-friendly rate control. Thus, it is necessary to obtain a new model, whose response function can be used to control the data rate of real-time flows over wireless links.

In order to address the issues introduced above, we present the analytical rate control (ARC) scheme for real-time multimedia flows over wireless links. Two key features of the analytical rate control approach construct our departure point.

- It decouples a single packet loss event from triggering rate control process.
- It achieves smooth variance in transmission rate, i.e., support for real-time multimedia transport.

ARC is an end-to-end connectionless analytical rate control scheme, which produces TCP-friendly flows with high throughput over wireless links. It is an equation-based rate control protocol whose control equation is obtained via modeling TCP behavior over lossy links by excluding its shortcomings. In order to obtain such an equation for rate control over wireless networks, we first establish an end-to-end path model based on the two-state Markov chain known as Gilbert's Model [15]. From the established model, we derive the driving equation for our analytical rate control scheme based on the desired behavior of a TCP source. The control equation is a function of the state probabilities of the path model and the RTT. The overall ARC protocol operation consists of two sequential periods, i.e., *probe period* and *steady period*. Since no link information is available in the beginning of the connection, the analytical rate control cannot be invoked. During this period, the ARC source calls the `Probe()` algorithm in order to determine the initial data transmission rate. At the end of this period, the source goes into *steady period* calling the `Steady()` algorithm. During the steady period, the data transmission rate S is controlled by using (21) as the required link information becomes available.

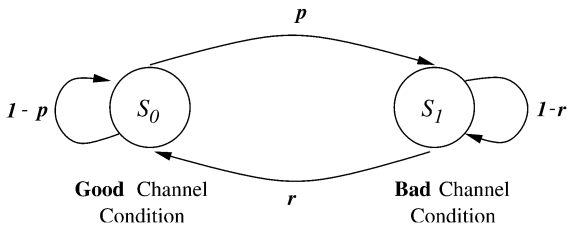

Fig. 1. Two-state Markov chain for packet loss process model for wireless channel.

The remainder of this paper is organized as follows. We establish the end-to-end path model which constructs the physical layer abstract for the derivation of rate control equation in Section II. In Section III, we present the model for desired TCP behavior over wireless links, which keeps core TCP rules and excludes its shortcomings over lossy links. In Section IV, we introduce the ARC protocol algorithm and its operation. The simulation experiment results and performance evaluation are discussed in Section V. Finally, the concluding remarks for the paper are presented in Section VI.

II. Communication Path Model

The end-to-end communication path is a concatenation of wireless and wired links. Hence, in order to obtain an accurate throughput model, it is necessary to capture channel behavior for the overall communication path. In Sections II-A–C, we first setup models for wireless channel and wired link individually, and then we establish the channel model for the end-to-end path.

A. Wireless Channel Model

Existing studies in the literature justify that a first-order Markov chain such as the two-state Markov model provides an adequate approximation to the error process of Rayleigh fading channel behavior [32], [35]. Hence, we model the wireless link part of the end-to-end connection path with the two-state discrete Markov chain known as Gilbert's Model [15], [27], [31] as shown in Fig. 1.

The two-state Markov chain can capture the bursty nature of the packet errors in fading channels. The model has two states as S_0 and S_1 representing *good* and *bad* channel conditions, respectively. When the state is S_0, i.e., good state, a packet is transmitted successfully. The transmission fails if the channel is in bad state, i.e., S_1. Success or failure of the transmission only depends on the current state. p and r are the state transition probabilities. The transition between two states occurs at each packet instant. For $P(S_0)+P(S_1)=1$ being steady-state state probabilities, the state transition matrix is given by

$$\mathbf{P_S} = \begin{bmatrix} 1-p & p \\ r & 1-r \end{bmatrix}. \tag{1}$$

The state transition probabilities depend on the error characteristics of the physical layer. The transition probabilities can be calculated from the channel characteristics such as the packet error rate (PER). The packet error rate (PER) of the wireless link is equal to the state probability $P(S_1)$ and depends on the transition probabilities [3]. Similarly, the probability of successful delivery of the packets over the wireless channel is, therefore, equal to the state probability $P(S_0)$.

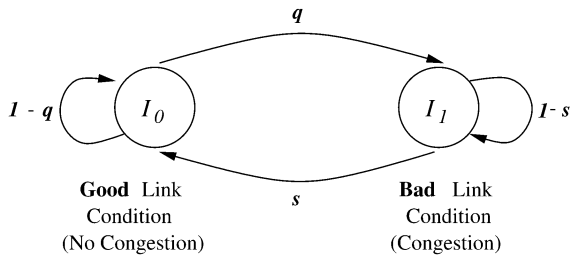

Fig. 2. Two-state Markov chain for packet loss process model for wired link.

B. Wired Link Model

Recent works on the measurements of unicast and multicast packet loss in the Internet have shown that the error process in the Internet can also be modeled with the two-state Markov chain [2], [8]. The state diagram with two states representing good and bad conditions, I_0 and I_1, is shown in Fig. 2.

Here, the bad wired link condition represents congestion situations. Hence, a packet is lost due to congestion, when the state is I_1. The packet is transmitted successfully otherwise. q and s are the state transition probabilities. For $P(I_0) + P(I_1) = 1$ being steady-state state probabilities, the state transition matrix is given by

$$\mathbf{P_I} = \begin{bmatrix} 1-q & q \\ s & 1-s \end{bmatrix}. \tag{2}$$

As in the wireless channel model, the packet loss rate due to congestion is equal to the state probability $P(I_1)$, and the probability of successful transmission over the wired link is, therefore, equal to the state probability $P(I_0)$.

C. End-to-End Path Model

Since the communication path consists of both wireless and wired parts, the model for the end-to-end path should capture behaviors of both portions of the path. Having individual packet error process models, we can construct the overall path model as a concatenation of the two models established in Sections II-A and B. The resultant four-state chain representing the end-to-end path is shown in Fig. 3.

In this model, a transition occurs between good (S_0I_0) and three bad states corresponding to bad wireless channel condition (S_1I_0), bad wired link condition (S_0I_1), and both (S_1I_1). S_1I_0 and S_0I_1 states represent only wireless link error and congestion conditions, respectively. When the state is S_0I_0, packet transmission is successful. Transmission fails if the state is in one of the bad states. Success or failure of the packet transmission solely depends on the current state. For $P(S_0I_0) + P(S_1I_0) + P(S_0I_1) + P(S_1I_1) = 1$ being the steady-state state probabilities, the state transition probability matrix is given as

$$\mathbf{P_{SI}} = \begin{bmatrix} (1-p)(1-q) & p(1-q) & q(1-p) & pq \\ r(1-q) & (1-r)(1-q) & rq & q(1-r) \\ s(1-p) & sp & (1-s)(1-p) & p(1-s) \\ rs & s(1-r) & r(1-s) & (1-s)(1-r) \end{bmatrix}. \tag{3}$$

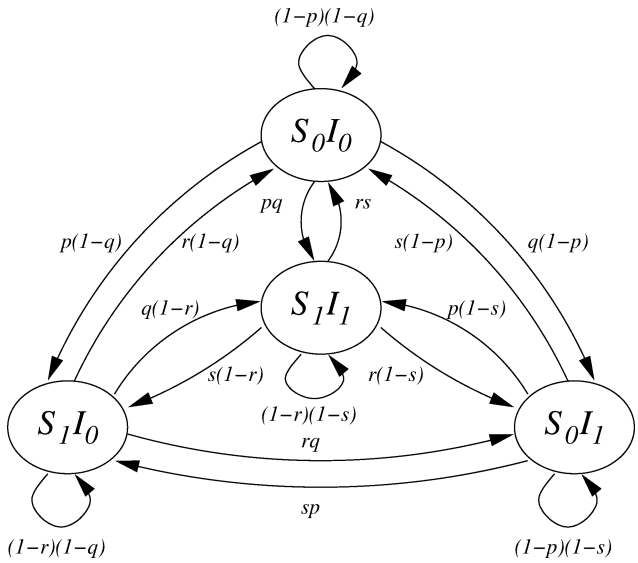

Fig. 3. Four-state Markov chain for packet loss in the end-to-end path including wireless and wired links.

In the four-state Markov chain shown in Fig. 3 representing the packet loss process for the end-to-end path, the total probability of packet loss, π, can be calculated by

$$\begin{aligned} \pi &= 1 - P(S_0I_0) \\ &= P(S_0I_1) + P(S_1I_0) + P(S_1I_1) \end{aligned} \tag{4}$$

where $P(S_0I_1)$ is the probability that the wireless link is error-free and congestion exists; $P(S_1I_0)$ is the probability that the wireless link is error-prone and no congestion exists; and $P(S_1I_1)$ is the probability that the wireless link is error-prone and congestion exists. Since the total packet loss probability is the sum of the probability that packet loss is due to congestion and the probability that packet loss is due to wireless error; by rearranging (4), π can also be expressed as

$$\pi = 1 - [1 - P(S_1)][1 - P(I_1)] \tag{5}$$

where $P(S_1)$ and $P(I_1)$ can be calculated by

$$P(S_1) = P(S_1I_0) + P(S_1I_1) \tag{6}$$

$$P(I_1) = P(S_0I_1) + P(S_1I_1). \tag{7}$$

Recall that $P(S_1)$ is the probability of being in the bad wireless channel state. On the other hand, the estimation of the state probability of $P(S_1)$ is a challenging task and there exist studies in the literature on the accurate estimation of the communication channel state probabilities and the packet loss classification techniques [18], [24]. For this reason, ARC protocol operation does not involve in direct estimation of $P(S_1)$, rather it uses the measured packet loss rate due to the wireless link at the link layer as an approximate to the state probability of the bad wireless channel. The details of the ARC protocol operation are presented in Section IV.

Note that the packet loss rate at the wireless link and π are both measurable quantities. Therefore, instead of calculating transition probabilities, i.e., p, r, q, and s, we use state probabilities of the end-to-end path model in developing our analytical rate control scheme, as presented in Section III.

III. Model for Desired TCP Behavior Over Wireless Links

Having established the model for the end-to-end communication path, the next step for our analytical rate control protocol is to derive the governing equation. The throughput equations used by the existing equation-based congestion control algorithms are based on the TCP behavior over reliable wired links [14], [21]. Therefore, they are not directly applicable to the wireless environments, since the TCP has poor performance in the error-prone wireless links. Thus, we need to obtain a new model such that it does not represent the classical TCP behavior over wireless links. The new model should, instead, capture the steady-state TCP behavior by excluding its drawbacks over lossy links. Therefore, the throughput model should also be consistent with the underlying channel model obtained for end-to-end path in Section II.

Using the model that captures the actual TCP behavior over wireless links is also not appropriate for this purpose. The models for the TCP behavior over lossy links do exist in literature [1], [17] and provide us with a better understanding of TCP's shortcomings. However, the use of these models to obtain a response function for equation-based rate control leads to experience the same throughput degradation with the TCP over wireless links. Therefore, we need to obtain a model for the desired TCP behavior over lossy links which excludes its drawbacks.

Most of the studies for TCP over wireless links aim to avoid TCP throughput degradation due to unnecessary rate decrease in case of link error losses by two major approaches. They either try to hide wireless link losses from the TCP source, or try to distinguish the loss reason and then act accordingly [4]. The former approach requires split or indirect solutions, most of which are not scalable and do not obey end-to-end semantics of congestion control. The latter method needs a very accurate and fast way of distinguishing the loss reason. What is common in these studies is that all aim to make the TCP source behave in a desired way, so that the TCP source over lossy link should

- perform no rate change due to wireless link error packet loss;
- decrease the rate when packet loss due to congestion occurs.

Therefore, we obtain a throughput model for such desired TCP behavior on top of the developed end-to-end path model. The resulting equation will then be used as the data rate controller for real-time multimedia traffic sources in the wireless networks. Note also that such rate control policy is suitable for the real-time multimedia flows which are loss-tolerant to a certain extent and have strict time constraints. Since the reliable transmission is of secondary importance due to the loss-tolerant nature of the real-time multimedia flows, it is better to keep transmitting time-sensitive packets with a controlled rate using a rate control protocol instead of backing off in case of wireless link errors.

Assuming that application always has data to send, TCP flow starting at $t = 0$ transmits $X(t)$ packets and achieves $T(t) = X(t)/t$ throughput in t time period. Therefore, the steady-state throughput, T, for such flow is

$$T = \lim_{t \to \infty} \frac{\overline{X}(t)}{t} \tag{8}$$

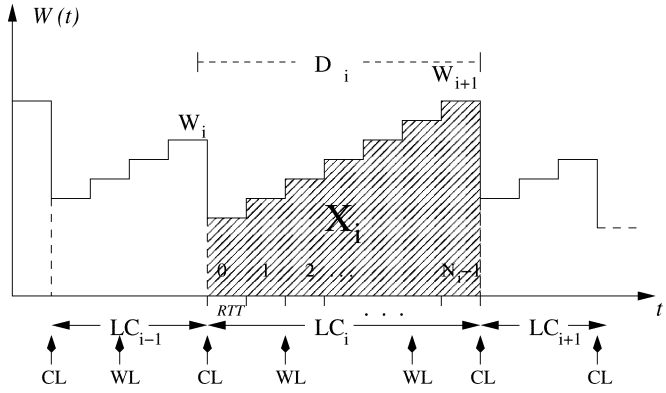

Fig. 4. Steady-state desired behavior of a TCP source in wireless link.

where $\overline{X}(t)$ is the expectation of the total number of packets transmitted in $[0, t]$ for $t \to \infty$.

The time dependency of the congestion window size, W, is shown in Fig. 4, where the packet loss events are due to either congestion loss (CL) or wireless link error (WL). We do not consider time-out events in our model. For packet loss rates up to 0.1, the model including only the triple-duplicate events is shown to be quite adequate approximation to the actual behavior in [21]. Furthermore, our goal is to capture desired source behavior over established path model and use its response function as a driving equation for rate control, rather than to obtain the most accurate analytical model for TCP source behavior over reliable links as in [21]. As it is shown in Fig. 4, there is no rate decrease at WL event arrivals, but there is at CL events. This is consistent with the desired TCP behavior that we want to model.

Each loss-free period, i.e., loss cycle LC_i, starts and ends with CL event arrival. W_i is the congestion window size at the beginning of each loss cycle just before the rate halving occurs. For the total number of packets transmitted X_i in the duration of D_i, the throughput achieved during LC_i is $T_i = X_i/D_i$. For each LC_i ending with X_i number of packets sent, the evolution of congestion window W_i can be assumed to be Markov regenerative process [19] with rewards X_i. Hence, the asymptotic throughput from the renewal theorem is expressed by

$$T = \frac{\overline{X}}{\overline{D}} \tag{9}$$

where $\overline{X}$ and $\overline{D}$ are the expectations that we need to obtain for X_i and D_i, respectively.

The congestion window at the jth RTT, $W_i[j]$, in a loss cycle i can be expressed as follows:

$$W_i[j] = \frac{W_i}{2} + j. \tag{10}$$

N_i is the number of RTTs in one loss cycle period. Thus, for $N_i - 1$ is the RTT index in which the next congestion loss event occurs, W_{i+1} is the congestion window at the end of the ith loss cycle and is equal to $W_i/2 + N_i$. Hence, the expectation of i.i.d. random variable W denoting congestion window size in LC_i can be expressed as

$$\overline{W} = 2\overline{N}. \tag{11}$$

The total number of packets X_i sent in the loss cycle LC_i lasting $N_i \cdot RTT$ duration can be calculated as follows:

$$X_i = \sum_{j=0}^{N_i-1} \left(\frac{W_i}{2} + j \right) = \frac{1}{2} \left(W_{i+1} + \frac{W_i}{2} - 1 \right) N_i. \quad (12)$$

Consequently, for mutually independent random variables of N_i and W_i, the mean of random variable X_i can be expressed as

$$\overline{X} = \frac{1}{2} \left(3\overline{N} - 1 \right) \overline{N}. \quad (13)$$

In a loss cycle LC_i, n_i packets are transmitted before the congestion loss occurs. Since a packet loss is detected in one RTT period, $W_i[N_i - 1]$ more packets are sent after the packet loss due to congestion occurs. Therefore, the total number of packet transmitted in LC_i can also be expressed as

$$X_i = n_i + W_i[N_i - 1]. \quad (14)$$

The random variable denoting the number of packets transmitted n_i is geometrically distributed with the unconditional probability of packet loss due to congestion $P(I_1)$. Hence, the probability that n packets are transmitted (including the lost one) is

$$P[n_i = n] = (1 - P(I_1))^{n-1} P(I_1). \quad (15)$$

Let π be the total packet loss probability and ω be the probability of packet loss due to wireless link errors. Recall that $\omega = P(S_1)$. Hence, it follows from (4)–(7) that $P(I_1)$ can be calculated as

$$P(I_1) = \frac{\pi - \omega}{1 - \omega} \quad (16)$$

where π is the total packet loss probability and ω is the probability of packet loss due to wireless link errors.

Hence, n_i gives the number of packets transmitted until a packet loss occurs by congestion which leads to rate throttle, rather than the wireless link error. Therefore, the mean of the total number of packets transmitted X_i can also be expressed as

$$\overline{X} = \overline{n} + \overline{W} \quad (17)$$

where the mean of random variable n_i can be obtained by using (15) and (16) as follows:

$$\overline{n} = \sum_k k P[n_i = k] = \frac{1 - \omega}{\pi - \omega}. \quad (18)$$

From (11), (13), (17), and (18), we can obtain the expectation of RTT count in a loss cycle, i.e., $\overline{N}$ as a function of state transition probabilities of the end-to-end path as follows:

$$\overline{N} = \frac{5}{6} + \frac{1}{6} \left[25 + 24 \left(\frac{1 - \omega}{\pi - \omega} \right) \right]^{1/2}. \quad (19)$$

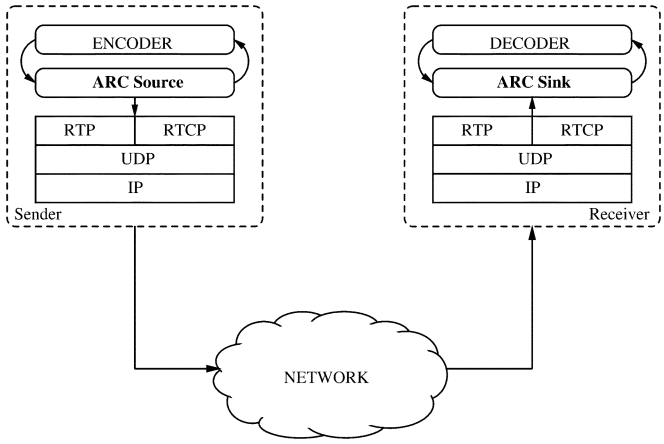

Fig. 5. ARC protocol structure.

The duration of a loss cycle LC_i is $D_i = N_i \cdot RTT$. Therefore, its expectation is calculated as

$$\overline{D} = \overline{N} \cdot RTT. \quad (20)$$

Thus, it follows from (13), (19), and (20) that the asymptotic throughput for the desired source behavior as a function of RTT and the state transition probabilities of the end-to-end path model can be expressed as

$$T = \frac{1}{4 \cdot RTT} \cdot \left(3 + \sqrt{25 + 24 \left(\frac{1 - \omega}{\pi - \omega} \right)} \right). \quad (21)$$

Note that the obtained throughput equation depends on state probabilities of the end-to-end path model developed in Section II. The throughput value as an output of (21) varies with RTT, π, and ω. The equation (21) gives the steady-state throughput that could be achieved if the TCP source would behave as desired. Hence, we use (21) as the driving equation for the analytical rate control operation of the ARC protocol.

IV. ARC: The Analytical Rate Control Scheme

ARC is an end-to-end analytical rate control protocol that uses the throughput function (21) developed in Section III. The objective of ARC is to perform rate control for real-time traffic over wireless links in order to produce TCP-friendly traffic flows while maintaining high throughput performance. Its equation-based approach provides a basis for multimedia traffic support in terms of time constraints such as delay and/or jitter bounds.

ARC can run on top of RTP/RTCP [26] and UDP as shown in Fig. 5 to provide real-time streaming support. ARC is not an ARQ protocol and the source does not perform retransmission due to tighter time constraints of the multimedia flows. While the rate control is mainly performed by the source, the receiver sends back an acknowledgment (ACK) for any received packet. If the RTP/RTCP is implemented, then these ACKs can be the receiver reports (RRs) that include the reception quality statistics such as the number of packets received, RTP timestamp, fraction lost, and cumulative number of packets lost. Hence, the ARC source can obtain the total packet loss rate π and RTT from the RTP receiver reports [26]. If RTP/RTCP is not used

for any reason, then data ACKs would be sufficient to acquire π and RTT.

The overall ARC protocol operation consists of two sequential periods, i.e., *probe period* and *steady period*. In the beginning of the connection, the information about the total packet loss rate π and the probability of packet loss due to wireless link errors ω are not available at the source. Hence, the data transmission rate, S, cannot be controlled by using (21) until the required information is obtained. During this period, the ARC source calls the `Probe()` algorithm in order to determine the initial data transmission rate. The *probe period* lasts $2 \cdot RTT$. At the end of this period, the data transmission rate S is set to the initial data rate S_{Init} and the source goes into the *steady period* calling the `Steady()` algorithm. The algorithm is presented in detail in Section IV-A.

During the steady period, the data transmission rate S is controlled by using (21) as the required link information becomes available. At each loss event arrival, the required link information, i.e., π, ω, and RTT are obtained and the data transmission rate S is set by using (21). The algorithm is presented in detail in Section IV-B.

A. Probe Period

The ARC source starts a new connection by calling the `Probe()` algorithm in the probe period, which lasts $2 \cdot RTT$. In [28], low-priority dummy packets are used to obtain available resource information at the beginning of the connection. Similarly, the ARC source transmits only one data packet at first and then low-priority *probe packets* in the first RTT with the target transmission rate, S_{Target}. Note that the target data rate S_{Target} could also be set by the application to achieve highest quality real-time streaming.

The low-priority probe packets can be distinguished by one of the eight bits of the IP *type of service* (TOS) field in the IP header. While there exists a considerable amount of ongoing research on the available bandwidth estimation techniques [16], the ARC protocol performs link probing only in the initial phase of the connection using low-priority *probe packets*. Currently, most of the commercial routers already have the *priority-queueing* capability based on the TOS field of the IP packet header [11]. While some routers in the Internet do not currently apply any priority policy, in the near future, the next-generation Internet will support QoS through the *Differentiated Service Model* (DiffServ) [7], which requires all routers to support multiple service classes with different service priorities and drop precedences. However, note that ARC does not specifically assume or require DiffServ architecture, rather a simple priority-queueing mechanism with two priority levels suffices for the *probe packet* mechanism.

The steps of the `Probe()` algorithm are shown in Fig. 6. For the ARC connection starting at $t = 0$, the ARC source sends probe packets at rate S_{Target} during $0 \leq t \leq RTT$. If there is congestion, low-priority probe packets are discarded first by the priority policy deploying routers. Otherwise, they are ACKed back to the sender yielding the initial resource availability information on the path. The ARC source counts the number of ACKs, n_{Ack}, received for the transmitted probe packets during

```
Probe()
  Send DATA_PACKET;
  While (t ≤ RTT)
    Send PROBE_PACKET with S_Target;
  end;
  If (t ≥ 2 · RTT) then
    Obtain n_Ack;
    S = max{1, n_Ack}/RTT;
  end;
    Steady();
end;

Steady()
  Foreach (LOSS_EVENT)
   Calculate π, ω;
   Obtain RTT;
   S = 1/(4·RTT) · (3 + sqrt(25 + 24((1−ω)/(π−ω))));
  end;
end;
```

Fig. 6. The `Probe()` and `Steady()` algorithms.

$RTT \leq t \leq 2 \cdot RTT$. Consequently, at the end of the second RTT period, i.e., $t = 2 \cdot RTT$, the ARC source sets its initial transmission rate to

$$S_{Init} = \frac{\max\{1, n_{Ack}\}}{RTT}. \tag{22}$$

B. Steady Period

The ARC source continues a connection by calling `Steady()` algorithm at $t = 2 \cdot RTT$. After the probe period, the steady-state operation of the ARC rate control depends on (21), and hence the total packet loss probability, π, the probability of packet loss due to wireless link ω, and RTT.

During this period, the ARC source uses (21) to control the data transmission rate S. To utilize (21) for this goal, the ARC source continuously obtains the packet loss rate π from RTP receiver reports (or it measures π within the sliding time window of τ using ACKs, if RTP/RTCP is not available). RTT can be obtained from the RTP/RTCP sublayer at the sender, or from the ACK packets. The probability of packet loss due to the wireless link ω is assumed to be obtained from the lower layers, i.e., the underlying MAC layer, which already has this information, or analytically from information available at the physical layer of the mobile node [10], [36]. Note that ω includes all wireless-link-related packet losses encountered by the underlying MAC layer due to bit errors, fading, and signal loss due to handoff or blackout. At each loss event arrival, the required variables π, ω, and RTT are obtained and the data transmission rate S is set by using (21). For the cases where the sender is not mobile station, the information regarding to the wireless portion of the end-to-end path, i.e., the probability of packet loss due to wireless link, ω, should be informed to the sender.

The overall operation of the `Steady()` algorithm of the ARC rate control scheme is shown in Fig. 6. During the steady period, at each loss event (`LOSS_EVENT`) the source obtains the required link information, i.e., π, ω, and RTT. Having all the required

information, the ARC source can then set its transmission rate S using the control equation, that is,

$$S = \frac{1}{4 \cdot RTT} \cdot \left(3 + \sqrt{25 + 24 \left(\frac{1-\omega}{\pi - \omega} \right)} \right). \qquad (23)$$

As a result, the connection starts with the probe period and then continues with the steady period. The rate control during the steady period is performed by the rate control function (23). At each loss event, the new π, ω, and RTT are obtained and then the rate control (23) is invoked.

On the other hand, the ARC rate control protocol can be in cooperation with an adaptive media encoding process. The adaptive encoder can be provided with the available bandwidth $S(t)$ estimated by the ARC, as shown in Fig. 5. Hence, the real-time multimedia encoder can then adapt its encoding rate $R(t)$ according to the controlled fair share of the network resources. In this way, the quality of the encoded multimedia can vary homogeneously and smoothly without leading to congestion and undesirable abrupt quality variations.

V. Performance Evaluation

The ARC is a rate control scheme for real-time traffic over wireless links. It is intended to achieve high throughput and multimedia support for real-time traffic flows while preserving fairness to the TCP sources sharing the same wired link resources. Hence, in order to investigate the performance of the ARC, extensive simulation experiments are conducted. We investigate ARC performance in terms of the throughput, goodput, fairness, and real-time multimedia support.

A. Throughput Performance

We simulate the topology given in Fig. 9, where N ARC sources are connected to the IP backbone via wireless access points. The corresponding N receivers are assumed to be fixed terminals in the network. For the simulation purposes, we set $N = 10$ and the capacity of all links $C = 1000$ packets/s.

We observed the throughput achieved by ARC, RCS, [28] and TFRC [14] sources for varying packet loss probability P_{Loss}. We choose RCS [28] for performance comparison, since it gives the best performance in lossy links to the best of our knowledge. Since TFRC is also an equation-based congestion control protocol which uses TCP response function as its control equation, we also compare ARC performance with TFRC. The connections passing through wireless channel are assumed to experience packet losses due to link errors with probability P_{Loss}.

In Fig. 7, we show the throughput of the ARC, RCS [28], and TFRC [14] protocols for different P_{Loss} values. Here, for low P_{Loss} values the same throughput is achieved by all protocols (ARC, RCS, and TFRC). However, for increasing P_{Loss} values, the throughput starts to decrease as expected but with different slopes. For example, for $P_{Loss} = 10^{-2}$, ARC improves throughput over TFRC and RCS by approximately 93% and 45%, respectively.

We also show the goodput achieved by the ARC, RCS, and TFRC for different P_{Loss} values in Fig. 8. A similar pattern with throughput is observed in this scenario. For $P_{Loss} = 10^{-2}$,

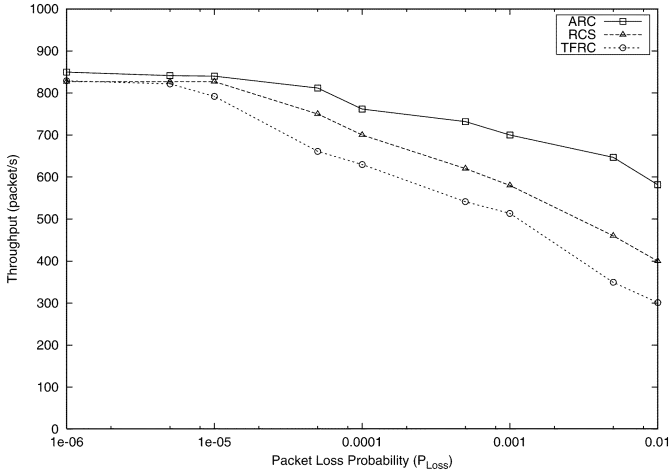

Fig. 7. Throughput performance of ARC, RCS, and TFRC for varying packet loss probability P_{Loss}.

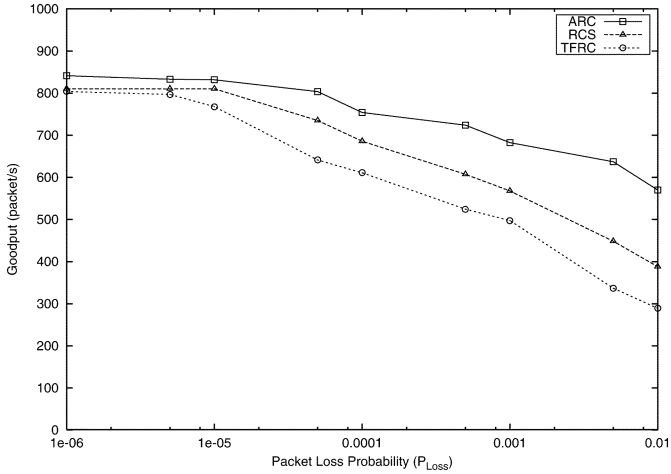

Fig. 8. Goodput performance of ARC, RCS, and TFRC for varying packet loss probability P_{Loss}.

ARC achieves goodput higher than TFRC and RCS by approximately 92% and 44%, respectively.

Note also that the difference between the goodput and the throughput achieved by the ARC sources increases as the wireless channel conditions get worse. In the worst case, all packets transmitted by the source can get lost in the wireless link for a certain period of time. Most of the media encoding techniques, on the other hand, are predictive and hence based on the previously encoded portion of the multimedia stream. Therefore, the encoding process can continue due to real-time constraints of the multimedia source even in the presence of excessive wireless link errors. However, advanced adaptive multimedia encoders that can adjust the encoding process according to the wireless channel conditions can be used at the application layer to improve the efficiency of the real-time multimedia encoding and hence transmission over the wireless links [30]. However, recall that ARC is a transport layer rate control protocol, which adjusts the data transmission rate such that the real-time multimedia flows use fair share of the link resources. Hence, the details of the efficient adaptive multimedia encoding based on the wireless channel state are beyond the scope of this paper.

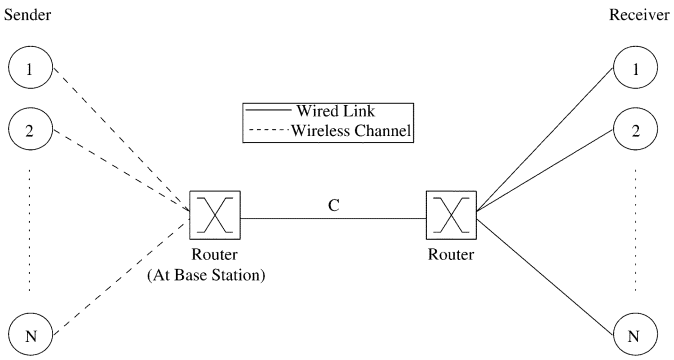

Fig. 9. Simulation scenario for throughput performance evaluation.

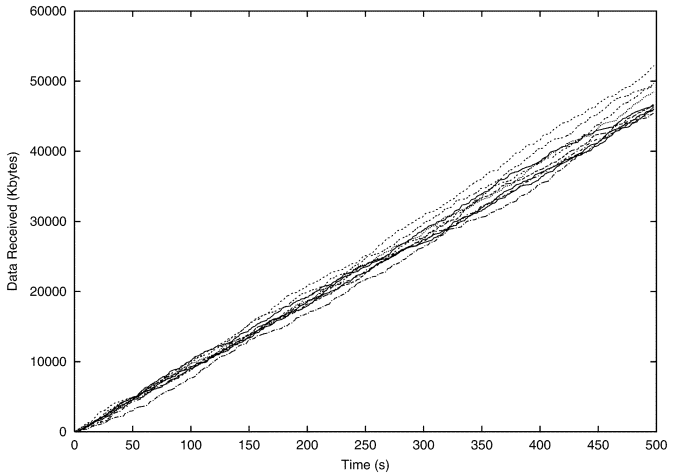

Fig. 10. Fairness among the ARC protocol sources: Data received at each ARC sink with respect to time.

B. Fairness

For the fairness performance of ARC protocol, we consider two different scenarios, i.e., homogeneous and heterogeneous fairness. For the homogeneous case, the fairness among ARC flows in sharing bottleneck is observed. For heterogeneous case, we explore the fairness of ARC protocol, where ARC sources share the bottleneck link with wired TCP sources.

1) **Homogeneous scenario:** In this case, we simulate the same topology shown in Fig. 9 with $N = 10$ and link capacity $C = 1000$ packets/s. Here, all of the senders deploy the ARC rate control scheme and share the same bottleneck. In Fig. 10, we show the data received at each ARC sink as a function of time t. It is observed from Fig. 10 that all sinks receive almost the same amount of data at any point in time during the simulation period. Therefore, we conclude that the ARC sources are all given a fair share of the network resources.
2) **Heterogeneous scenario:** In this case, we simulate the topology shown in Fig. 11, where $N = 10$ ARC connections and $N = 10$ TCP-NewReno [13] connections share the same bottleneck with capacity $C = 1000$ packets/s. Here, the ARC sources are connected to the bottleneck via a wireless channel (dashed lines in Fig. 11) experiencing $P_{Loss} = 10^{-3}$ due to wireless link errors. TCP sources are connected to the bottleneck via wired links (solid lines in Fig. 11), hence they do not experience any wireless

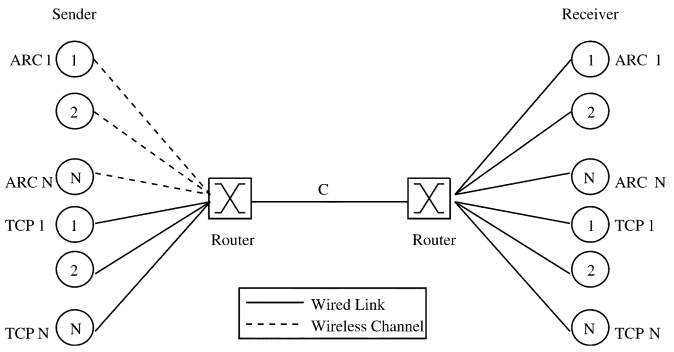

Fig. 11. Simulation scenario for heterogeneous fairness evaluation.

link errors. The reason for simulating such topology is basically that we are interested in fairness performance of ARC to the wired TCP sources sharing the same bottleneck link.

Let Φ be the fairness between the ARC and TCP sources, where Φ is the ratio of the average throughput of ARC (T_{ARC}) and TCP (T_{TCP}) connections, i.e.,

$$\Phi(t) = \frac{T_{ARC}(t)}{T_{TCP}(t)}. \tag{24}$$

Therefore, if $\Phi(t) \approx 1$, then ARC and TCP sources are given a fair share of bottleneck. In Fig. 12(a) and (b), we show the average data received by the ARC and TCP receivers and their ratio $\Phi(t)$ as a function of time, respectively. As shown in Fig. 12(a), the wired TCP sources are always given a higher share of the bottleneck than the ARC sources. In Fig. 12(b), we have $\Phi(t) < 1$ for all $t < 0$, concluding that ARC preserves fairness to the wired TCP sources sharing the same wired link resources while significantly enhancing network utilization efficiency.

We have also performed simulation experiments using the same simulation environment shown in Fig. 11, where, in this case, TCP sources also traverse wireless links as ARC sources. The objective of this set of simulation experiments is to observe how ARC sources maintain fairness after temporary wireless link errors are over, and consequently, how TCP sources recover from unnecessary rate decreases and start to obtain the fair share of the link resources.

Here, we introduce wireless link errors with $P_{Loss} = 10^{-3}$ at $t = 25$ s for a duration of 20 s. As it is observed in Fig. 12(c), TCP sources have higher share of the bottleneck until the wireless link errors are introduced. After $t = 25$ s, ARC sources start to obtain higher link resources. During $25\text{ s} < t < 45$ s, ARC sources utilize higher bandwidth than TCP sources. This is because TCP sources unnecessarily perform rate decrease due to misinterpretation of packet losses due to wireless link errors as congestion losses as discussed in Section III. Note that effective loss-labeling schemes such as in [6] can improve the TCP performance in wireless networks by enabling TCP sources to more accurately identify packet loss reasons. In this way, the transient unfairness of ARC sources

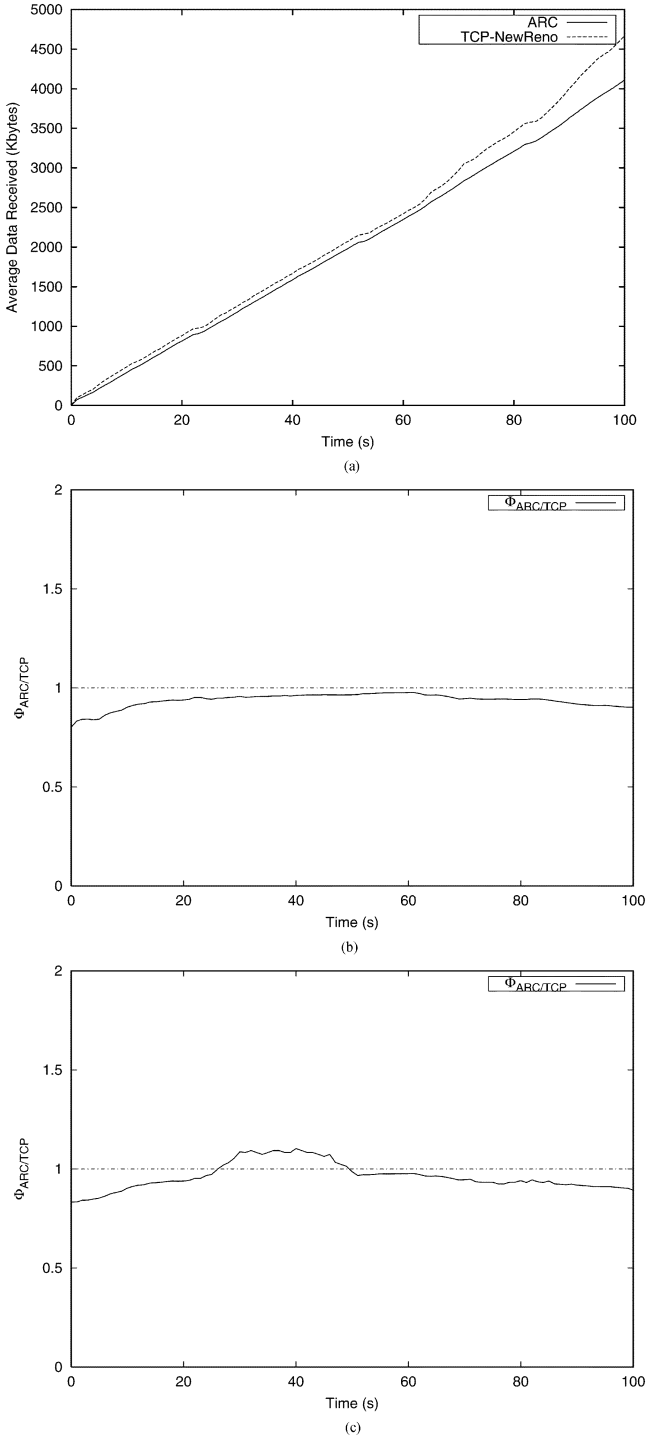

Fig. 12. Fairness to wired TCP sources. (a) Average data received at each ARC and TCP-NewReno sink. (b) Ratio of the received data with respect to time. Fairness to wireless TCP sources. (c) Ratio of the received data with respect to time for the cases where TCP sources also traverse wireless links.

to TCP sources observed in Fig. 12(c) due to the inherent shortcomings of TCP can be further minimized.

However, as shown in Fig. 12(c), as the wireless link errors are over, the link share of the ARC sources decreases. This is because TCP sources start to recover from the erroneous rate decrease they performed in case of wireless link errors. As shown in Fig. 12(c), after $t = 50$ s, $\Phi < 1$. Therefore, ARC sources can maintain the fairness after the wireless link errors are over and hence TCP sources can recover their fair share while competing with ARC sources for the same bottleneck resources.

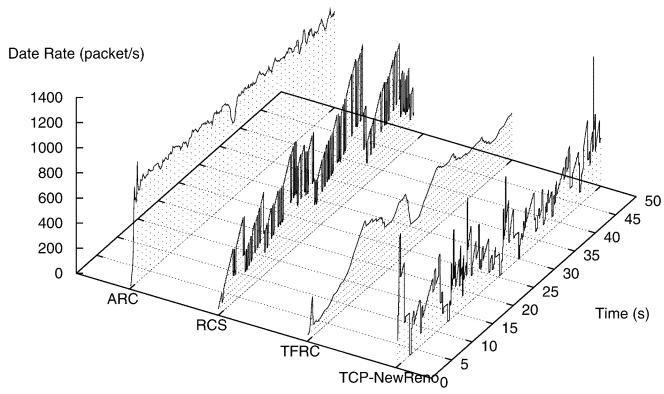

Fig. 13. Transmission rate variation of ARC, RCS, TFRC, and mobile TCP-NewReno sources with time.

C. Multimedia Support

In order to investigate ARC performance in terms of multimedia time-constraints, we choose to observe its rate change pattern and delay variation, i.e., jitter. One of the most important properties of equation-based rate control is its smoothly changing data transmission rate. The transmission rates of ARC, RCS [28], TFRC [14], and TCP-NewReno [13] sources are shown in Fig. 13 as a function of time.

In this scenario, we perform simulation experiments with ARC, RCS, TFRC, and TCP-NewReno sources using the same simulation scenario given in Fig. 9. In the experiments, the sources are in wireless domain experiencing packet loss probability $P_{Loss} = 10^{-2}$. As shown in Fig. 13, the most frequent rate change is experienced by the mobile TCP-NewReno source. This is because the mobile TCP source interprets every packet loss as a congestion indication, hence it performs unnecessary rate decreases. TFRC also provides very smooth rate change, however, it also cannot distinguish wireless link error losses than congestion losses. Therefore, TFRC also experiences unnecessary rate decrease during connection time. Since RCS follows the conservative TCP rule, i.e., rate halving at packet loss, and then tries to recover from this rate change by using dummy packets [28], the RCS source also experiences more frequent rate change compared to the ARC source, as shown in Fig. 13.

We also investigate the jitter experienced by ARC, RCS [28], TFRC [14], and TCP-NewReno [13] flows. The jitter experienced by each packet transmitted by the ARC, TFRC, and TCP sources are shown in Fig. 14(a)–(d), respectively. In Fig. 14(d), it is observed that the packets sent by the TCP source always experience much higher jitter than the packets sent by TFRC, RCS, and ARC. This is because TCP performs rate throttle at each packet loss hence resulting in high delay variation. RCS [28] also performs rate halving at each packet loss; however, since it recovers from unnecessary rate halving by using dummy packets, RCS source experiences less jitter than TCP source. TFRC also suffers from the wireless link errors hence makes unnecessary rate changes. As observed in Fig. 14(a)–(c), the jitter

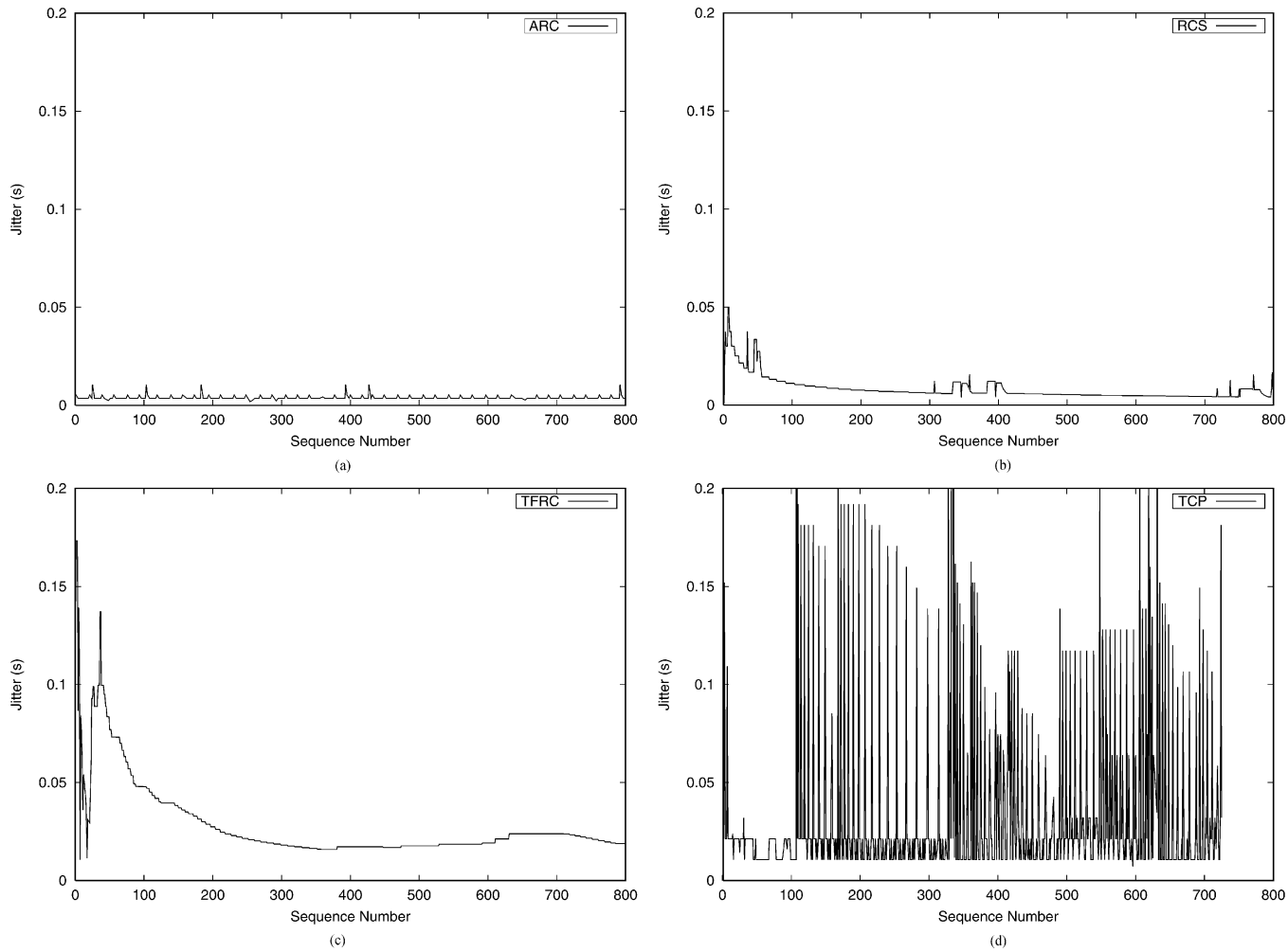

Fig. 14. Jitter experienced by each packet sent by (a) ARC, (b) RCS, (c) TFRC, and (d) TCP-NewReno sources.

experienced by ARC, RCS, and TFRC packets is mostly around 5, 10, and 25 ms, respectively. Thus, ARC outperforms RCS, TFRC, and TCP in terms of jitter-bounds of the real-time multimedia transmission over wireless links.

VI. Conclusion

Next-generation wireless Internet (NGWI) is expected to provide a wide range of services including real-time multimedia to mobile users. The real-time multimedia traffic transport requires rate control deployment to protect the shared Internet from unfairness and further congestion collapse. However, the existing solutions are mostly for the wired Internet and hence they do not address the challenges in the wireless environments.

In this paper, we introduced the ARC, a new analytical rate control protocol for real-time multimedia traffic over wireless networks. ARC is an end-to-end analytical rate control protocol which is designed to produce TCP-friendly real-time traffic flows while maintaining high throughput performance in wireless networks. The desired TCP behavior over wireless links is captured based on the end-to-end path model developed. The asymptotic throughput equation is derived and used as the control equation for the ARC rate control scheme. Its equation-based approach provides the basis for multimedia traffic support in terms of time constraints such as delay and/or jitter bounds. Simulation results show that ARC significantly improves the rate control throughput performance over wireless links and provides multimedia traffic support without penalizing TCP sources sharing same path.

References

[1] A. A. Abouzeid, S. Roy, and M. Azizoglu, "Stochastic modeling of TCP over lossy links," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 2000, pp. 1724–1733.

[2] E. Altman, K. Avrachenkov, and C. Barakat, "TCP in presence of bursty losses," in *Proc. ACM SIGMETRICS*, June 2000, pp. 124–133.

[3] S. Aramvith, I. M. Pao, and M. T. Sun, "A rate-control scheme for video transport over wireless channels," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 569–580, May 2001.

[4] 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, pp. 756–769, Dec. 1997.

[5] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for Internet hosts," in *Proc. ACM SIGCOMM*, Sept. 1999, pp. 175–187.

[6] D. Barman and I. Matta, "Effectiveness of loss labeling in improving TCP performance in wired/wireless networks," in *Proc. IEEE ICNP*, Nov. 2002, pp. 2–11.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," Network Working Group, RFC 2474, Dec. 1998.

[8] J. Bolot, S. Parisis, and D. Towsley, "Adaptive FEC-based error control for Internet telephony," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 1999, pp. 1453–1460.
[9] 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.
[10] A. Chockalingam, M. Zorzi, L. B. Milstein, and P. Venkataram, "Performance of a wireless access protocol on correlated Rayleigh-fading channels with capture," *IEEE Trans. Commun.*, vol. 46, pp. 644–655, May 1998.
[11] (2003) Cisco Systems. [Online]. Available: http://www.cisco.com
[12] A. J. Cobb and P. Agrawal, "Congestion or corruption? A strategy for efficient wireless TCP sessions," in *Proc. IEEE Symp. Computers and Communications*, 1995, pp. 262–268.
[13] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," Network Working Group, RFC 2585, Apr. 1999.
[14] 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.
[15] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, pp. 1253–1265, Sept. 1960.
[16] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 295–308.
[17] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Networking*, vol. 5, pp. 336–350, June 1997.
[18] J. Liu, I. Matta, and M. Crovella, "End-to-end inference of loss nature in a hybrid wired/wireless environment," in *Proc. Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'03)*, Sophia-Antipolis, France, Mar. 2003.
[19] D. Logothetis, K. S. Trivedi, and A. Puliafito, "Markov regenerative models," in *Proc. Int. Computer Performance and Dependability Symp.*, Erlangen, Germany, 1995, pp. 134–143.
[20] C. Parsa and J. J. Garcia-Luna-Aceves, "Differentiating congestion vs. random loss: A method for improving TCP performance over wireless links," in *Proc. IEEE WCNC*, vol. 1, 2000, pp. 90–93.
[21] 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, pp. 133–145, Apr. 2000.
[22] 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, pp. 18–32, Mar. 2001.
[23] 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.
[24] K. Salamatian and S. Vaton, "Hidden Markov modeling for network communication channels," in *Proc. ACM SIGMETRICS*, June 2001, pp. 92–101.
[25] N. K. G. Samaraweera, "Non-congestion packet loss detection for TCP error recovery using wireless links," *IEE Proc.—Commun.*, vol. 146, no. 4, pp. 222–230, Aug. 1999.
[26] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport for real-time applications," Network Working Group, RFC 1889, Jan. 1996.
[27] F. Swarts and H. C. Ferreira, "Markov characterization of digital fading mobile VHF channels," *IEEE Trans. Veh. Technol.*, vol. 43, pp. 977–985, Nov. 1994.
[28] 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.
[29] M. Vojnovic and J. Y. Le Boudec, "On the long-run behavior of equation-based rate control," EPFL, Lausanne, Switzerland, Tech. Rep. IC/2002/06 EPFL, 2002.
[30] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*. New York: Prentice-Hall, 2002.
[31] H. S. Wang and N. Moayeri, "Finite-state Markov channel: A useful model for radio communications channels," *IEEE Trans. Veh. Technol.*, vol. 44, pp. 163–171, Feb. 1995.
[32] H. S. Wang, "On verifying the first-order Markovian assumption for a Rayleigh fading channel model," *IEEE Trans. Veh. Technol.*, vol. 45, pp. 353–357, May 1996.
[33] J. Wroclawski, "The use of RSVP with IETF integrated services," Network Working Group, RFC 2210, Sept. 1997.
[34] Y. R. Yang, M. S. Kim, and S. S. Lam, "Transient behaviors of TCP-friendly congestion control protocols," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 2001, pp. 1716–1725.
[35] M. Zorzi, R. R. Rao, and L. Milstein, "On the accuracy of a first-order Markov model for data transmission on fading channels," in *Proc. IEEE ICUPC*, 1995, pp. 211–215.
[36] M. Zorzi, R. R. Rao, and L. B. Milstein, "ARQ error control for fading mobile radio channels," *IEEE Trans. Veh. Technol.*, vol. 46, pp. 445–455, May 1997.

Özgür B. Akan (S'02) 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.

Ian F. Akyildiz (M'86–SM'89–F'95) 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 Communication 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."