RAAR: A TCP-friendly Congestion Control Mechanism of Transporting Multimedia Traffic in Internet

Yan Hu
Network Research Department, Institute of Computing Technology, Chinese Academy of Sciences
Beijing, P. R. China
huyan@ict.ac.cn

Guangzhao Zhang and Wanqing Tu
Dept. of Electronics & Communication Engineering
Zhongshan University
Guangzhou, P. R. China
isszgz@zsu.edu.cn, tuwanqing@163.net

Abstract—**This paper proposes a unicast mechanism of Rate Adaptation At Receivers called RAAR. It can be used to transport multimedia traffic. UDP and TCP dominate in current Internet. Neither TCP nor UDP can be used by multimedia traffic, because the TCP reduces the sending rate in half in response to a single packet drop and UDP has no congestion control mechanism. RAAR ameliorates GAIMD at receivers and has good smoothness of sending rate and fairness with competing TCP flows. RAAR is simple to implement. Our simulations show that the performances of RAAR are better than TFRC obviously. RAAR is also a promising scheme of development for congestion control of multicast multimedia traffic, because it is not a per-packet acknowledgement mechanism and its rate adaptation is implemented at receivers.**

Keywords—TCP-friendly; AIMD; rate adaptation; congestion control

I. INTRODUCTION

A great many video and audio flows, also called multi-media traffic flows or multimedia real-time flows, have been transmitted in the Internet. It is expected that multimedia streaming traffic will increase rapidly, and will soon make up a significant portion of the total Internet bandwidth in the coming future.

Multimedia flows are characterized by: 1) delay- sensitive: a transmission session, which has short a delay and a low jitter, is expected by multimedia traffic; 2) information-huge: generally, the information in the multimedia files is much greater than the one in the common data files; 3) high transmission rate; 4) high tolerance to the code-error: very low code-error probability is required in data flows, while a higher code-error probability can be acceptable in multimedia flows, because it only reduce the playback quality which can be accepted by the users. However, the current Internet does not attempt to guarantee an upper bound on end-to-end delay or lower bound on available bandwidth. As a result, the quality of delivered service to real-time applications is neither controllable nor predictable. So in the best-effort network as the Internet, we can't guarantee the QoS of the multimedia traffic. Lack of support for QoS has not prevented rapid growth of multimedia traffic and this is expected to continue.

At present, there is no end-to-end congestion control mechanism in most of real-time multimedia applications, or those flows are not TCP-friendly. They would go against the Internet if there were a great deal of such flows transmitted. A feasible way to address the issue is using the RSVP [21] or Differentiated Service [4]. Even though those kinds of services could be spread in the Internet, many users still want to get cheaper real-time services, and of course, the best-effort service is the cheapest one. If users are in the same service level in the networks supporting the Differentiated Service, the services shared by them are also a kind of best-effort ones. So we can see it is very significant for us to study the transmission protocol of real-time multimedia flows in the best-effort network.

Transmitting multimedia traffic using UDP that has no congestion control will lead to some serious problems. The transmission in the best-effort network in despite of congestion state tends to cause serious packet losses that make the utilization of the network very low. The more trouble situation is that: the lost packets, which cannot reach the destination forever, occupy most of bandwidth, while the senders send packets repeatedly regardless of the network congestion. Finally, the network meltdown happens. At the same time, the goodputs of the multimedia flows are also very low [3], [8]. In the current Internet, where 95% throughput belongs to TCP, TCP throughput will be decreased greatly due to the kind of unfairness. Hence, in order to reduce the UDP loss and increase the bandwidth utilization, we need provide the congestion control mechanism for UDP to transmit those multimedia flows.

TCP is the dominant transmission protocol in the Internet, and the current stability of the Internet depends on its end-to-end congestion control that uses an Additive Increase Multiplicative Decrease (AIMD) algorithm [6]. However, the TCP congestion control only is appropriate for applications such as bulk data transfer and not for the transfer of multimedia traffic, because the behaviors of TCP, which halve the sending rate in responsible to a single congestion indication, will cause the intensive jitter of transmission rate and noticeably reduce the user-perceived quality. As for the asymmetric network (such as, wireless network, cable modems, ADSL, and satellite network), it is more serious.

This research was done when the first author was at the Department of Electronics & Communication Engineering, Zhongshan University, Guang-zhou City, Guangdong Province, P. R. China.

Because of lack of bandwidth on the reverse links in those networks, TCP that feedbacks ACK packet on receiving each data packet is not appropriate. In the asymmetric network, delays and packet losses occurring on the reverse links severely degrade the performance of existing round trip based protocol such as TCP. TCP is also ill suited for the multicast multimedia traffic. In a large-scale multicast involving many receivers (10K to 1M receivers), frequent feedback sent directly to the sender causes implosion, at the same time those senders' burden becomes greater and greater.

In the shared network such as the Internet, in order to avoid the congestion and improve the network utilization, all of the end system (including the real-time one and the non real-time one) should decrease their transmission rates whenever there are congestions, and should increase them as no congestion. So an ideal multimedia transmission scheme should have such a rate adaptation mechanism, while the inter-protocol fairness must be considered, i.e., the variant protocol flows that coexist in the same link can share bandwidth fairly. Because the dominant traffic in the Internet is based on TCP, such as e-mail, FTP, and Web etc., in order to meet the demand of the fairness among the protocols, transmission protocols of the multimedia flows should make the throughput of their traffic flows approximately equal to the TCP's. However, as we said before, TCP is not suitable for transmitting multimedia flows, so some improvement on TCP must be made, and the TCP-friendly idea was proposed. TCP-friendly is that a real-time multimedia flow should obtain approximately the same average bandwidth over the timescale of a session as a TCP flow along the same path under the same conditions of delay and packet loss [11]. Certainly, it is only defined in view of fairness. As an excellent multimedia transmission protocol, it should also consider the characteristics of the multimedia traffic flows (see the previous part) at the same time. This kind of TCP-friendly transmission protocol is an ideal multimedia transmission scheme, if it satisfies the both requirements.

To design a TCP-friendly congestion control protocol, several elementary targets should be achieved: 1) fairness: small variations over the sending rates of competing flows such as TCP flows, 2) smoothness: small sending rate variations over time for a particular flow in a stationary environment, 3) responsiveness: fast deceleration of protocol sending rate when there is a step increase of network congestion, and 4) aggressiveness: fast acceleration of protocol sending rate to improve network utilization when there is a step increase of available bandwidth [24].

The balance of this paper is organized as follows. Some proposed TCP-friendly schemes in literature are introduced in Section 2. Our RAAR scheme is described in Section 3. In Section 4, we give the simulation results in all kinds of configuration and the metrics to evaluate the performances of TCP-friendly protocol. Our conclusions and future work are in Section 5.

II. RELATED WORK

The proposed TCP-friendly congestion schemes in literature fall into two major categories: AIMD-based [2], [7], [10], [15], [16], [17], [20], [25] and formula-based [9], [11], [14], [19].

TCP congestion control algorithms are based on the window or rate adaptation principle of Addition Increase Multiplicative Decrease (AIMD) [6], which may be expressed as:

$$I : w_{t+R} \leftarrow w_t + \alpha;\ \alpha > 0$$
$$D : w_{t+\delta t} \leftarrow \beta \cdot w_t ; 0 < \beta < 1$$

where I refers to the increase in window as a result of receipt of one window of acknowledgements in a RTT and D refers to the decrease in window on detection of a loss by the sender, w_t the size of the window at t, R the round-trip time of the flow, and α, β are constants. In [6], [25], the authors discussed the stability and the fairness of those algorithms. [2] generalized the AIMD rules, introduced and analyzed a class of nonlinear control algorithms called binominal algorithms. They concluded a $k + l$ rule, which represents a fundamental tradeoff between probing aggressiveness and the responsiveness of window reduction.

The congestion control mechanism of Rate Adaptation Protocol (RAP) [15] is implemented at senders. The RAP source sends data packets with sequence numbers, and a RAP sink acknowledges each packet, providing the end-to-end feedback. It is a transmission mechanism of rate-based congestion control. If no congestion is detected, its source periodically increases the transmission rate. If congestion is detected, it immediately decreases the transmission rate. In order to decrease the oscillation of the transmission rate, RAP uses a fine gain rate adaptation scheme, which can smooth the rate to some degree. It is not necessary for multimedia flows to acknowledge each data packet. It increases workload of the network and not appropriate for asymmetric networks and multicast. Although a fine gain rate adaptation scheme is used in RAP, its transmission rate is still too oscillatory to transport real-time multimedia flows.

In [16], the authors proposed a protocol called TEAR (TCP Emulation At Receivers) that shifts most of flow control mechanisms to receivers. In TEAR, a receiver does not send to the sender the congestion signals detected in its forward path but rather processes them immediately to calculate its own appropriate receiving rate. TEAR doesn't use the per-packet acknowledgement scheme like TCP and RAP, so TEAR, which applies some form of weighted averaging over rate samples taken over W = 8 epochs in the past to smooth the transmission rate, can be used for either unicast or multicast of real-time multimedia traffic in asymmetric networks. However, the TEAR protocol is complicated to implement.

In recent years, there is a lot of research on modeling TCP throughput. These models are able to predict TCP throughput over a wider range of parameters such as loss rates. In [9], Floyd etc. apply those results to propose a mechanism of equation-based congestion control for unicast of multimedia traffic, which is called by TFRC (TCP-friendly Rate Control). Owing to the fault of those own models, when the packet loss rate is very high, the performances of TFRC are not acceptable [16], [24].

III. POROPOSED RAAR PROTOCOL

We propose a novel TCP-friendly approach to flow control called Rate Adaptation At Receivers (RAAR) for unicast multimedia streaming and it can also be upgrade to multicast. Our design goal is to develop a flow control protocol that: 1) can fairly share the bandwidth with the competing TCP; 2) has good smoothness of sending rates, which is suitable for transmitting multimedia traffic; 3) can avoid the feedback implosion, which is hard to upgrade to multicast; 4) is suitable for not only the traditional symmetric networks but also the emerging asymmetric networks, such as satellite communication networks.

The rate-based adaptation congestion control of RAAR simulates the one of TCP, so it can be TCP-friendly. Unlike TEAR and TCP, RAAR is a rate-based control rather than window-based one. Using some smoothness function of rate, RAAR can have good smoothness of rate. Our protocol is not a per-packet acknowledgement mechanism and its rate control is achieved at receivers, so it can avoid the feedback implosion and lessen workload of the senders.

Our RAAR protocol mainly derives from General Additive Increase Multiplicative Decrease (GAIMD) [25] algorithm. However, RAAR shifts the rate adaptation congestion control mechanism to the receivers while RAP or GAIMD implements the mechanism at its senders.

Firstly, the sender of RAAR sends data packets at some initial rate set in advance. The receiver estimates a sending rate using the GAIMD algorithm, and sends ACK packets to report the sending rate to the sender if one of following two conditions are met: 1) the latest sending rate estimated by the receiver is less than the current sending rate at the sender; 2) the RTT timer at the receiver expires. We define a *round* in RAAR protocol. After the sender receives an ACK packet, it immediately updates its sending rate using the rate in the ACK packet and sends those latter data packets at the new rate, which also means a new round has begin. This round will not end until the sender receives another new ACK packet. The next round begins on the last round ending. It is illustrated in Fig. 1, which shows that there are continuous alternant rounds in RAAR. RAAR can be divided into two functionalities, i.e., sender functionality and receiver functionality. We will discuss them in detail as follows.

A. *Sender Functionality*

A.1 Function of Sending Data Packet at Specified Rate

One of the principles of RAAR is that the sender should be as simple as possible. Thus, we shift almost all workload to the receiver to alleviate the burden of the sender.

A timer is needed to control the sending rate at the sender. Supposing the current sending rate is *rate_*, and the packet size is *pktSize_*. Thus, we can start a timer with a length of *pktSize_/rate_* after a packet has been sent out. The next packet should be sent as soon as the timer expires. So in this way, we can guarantee the sender sends the packet at the rate of *rate_*.

The difference from TCP [18] or RAP is that RAAR doesn't acknowledge each data packet, while only sends an

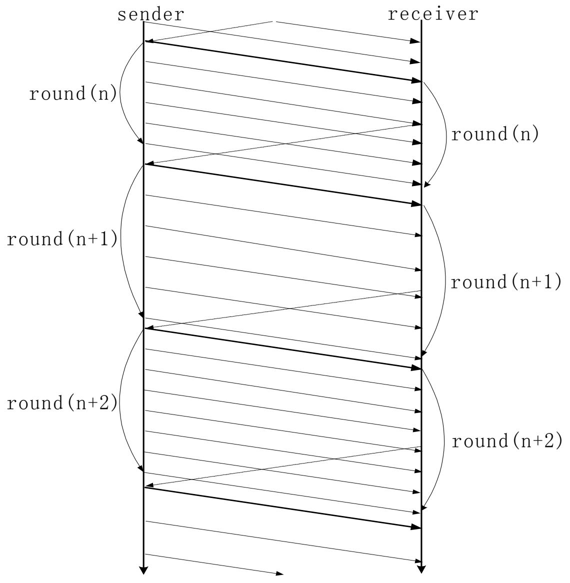


Figure 1. RAAR protocol

ACK packet when each round ends (see Fig. 1). This mechanism is well-suited for transmission of multimedia traffic flows, because multimedia traffic can tolerate error-rate to some degree as the result of its function to recover error-code at receivers. The feedback implosion [1], [20] is avoided in the mechanism, so it, an acknowledging per-round mechanism, is also a promising avenue of development for multicast traffic and asymmetric networks.

A.2 Function of Decreasing the Sending Rate for Timeout

In any networks, it is unavoidable for packet losses, so does the ACK packet of RAAR. Although the loss probability of the ACK packets is very low in RAAR, the ACK packets are very important to the rate adaptation of RAAR. Then we must propose a mechanism how RAAR is responsive to the ACK packet losses. The sending rate should be decreased correspondingly as the ACK packets are lost. When there is no ACK packet loss, the senders should receive an ACK packet every the time of $t \le 2RTT$. The sender should decrease the sending rate if an ACK packet loss occurs. We propose the detail mechanism as follows. The sender gets the values of retransmit timeout *RTO* and *rate_* (the both parameters are estimated by the receiver, see Section 3.B.1 and 3.B.3), and then sends a data packet at the rate of *rate_*. If the sender does not receive an ACK packet after the time of 2*RTO*, it should immediately update the sending rate as $\beta \times rate\_$ (where β is the multiplicative decrease factor of GAIMD, see Section 3.B.1)

B. *Receiver Functionality*

B.1 Implementation of Improved GAIMD Algorithm at Receivers

The leading goal in RAAR is to implement the GAIMD algorithm at receivers. Two parameters, i.e. α (α>0) and β (0<β<1), are defined as: in the congestion avoidance state, the window size is increased by α per window of packets acknowledged and it is decreased to β of the current value whenever there is a triple-duplicate congestion indication. As for TCP, α is one and β is 0.5. [25] proposed a simple relationship between α and β for a GAIMD flow to be TCP-friendly, that is, for the GAIMD flow to have approximately the same throughput as a TCP flow. The relationship between α and β to be TCP-friendly is

$$\alpha = \frac{4(1-\beta^2)}{3} \tag{1}$$

Our RAAR modifies the GAIMD algorithm when it is implemented at receivers. The RAAR protocol consists of two states: 1) slow-start and 2) congestion-avoidance state. The slow-start state is a process of detecting the available bandwidth in the network. The RAAR flows can detect its available bandwidth using the slow-start algorithm. If RAAR continues to increase its sending rate after it has utilized the available bandwidth, there will be packet losses. Thus, RAAR changes the state to the congestion-avoidance state, a process of the dynamic balance that the throughput of a RAAR flow fluctuates about a value of its available bandwidth.

Like TCP, after a RAAR session has been set up and its first data packet arrives at a receiver, The RAAR enters into the slow-start state. Hereafter, in order to detect the available bandwidth in current network congestion state as soon as possible, the receiver updates the current rate *rate_* as $rate\_ + pktSize\_ / RTT$ (where *pktSize_* refers to the size of data packets and *RTT* round trip time) whenever it receives a data packet. Just as Fig. 1 illustrates, a timer with a length of *RTT* starts when a new round begins. Only if the timer expires or *rate_* is less than the current sending rate carried by the latest data packet, the receiver reports the sender the *rate_* immediately, that is, the receiver sends an ACK packet with the *rate_* value to the sender.

The receiver decides whether there is a loss event, when a data packet arrives at the receiver. We will define the loss event in Section 3.B.2. RAAR changes into the congestion-avoidance state, if its receiver detects a loss event. In the state, RAAR uses the improved GAIMD algorithm.

If a packet arrives at the receiver and there is no a loss event, the sending rate value *rate_* can be updated as follows:

$$rate\_ \leftarrow rate\_ + \frac{\alpha \cdot \delta}{wnd\_} \times \frac{pktSize\_}{RTT} \tag{2}$$

where *wnd_* is the number of data packets received by the receiver in a round, *RTT* is the smoothed round-trip time measured by the receiver, and δ is a factor measured by experiment. Using simulations, we find that δ can be set as a value between 1/5 and 1/10. In all of simulations in this paper, we let δ=1/6. As we have known, at the congestion avoidance state, the sender of TCP (or GAIMD) increases its congestion window as follows whenever receiving an ACK packet,

$$cwnd\_ \leftarrow cwnd\_ + \frac{\alpha}{cwnd\_} \times pktSize\_$$

where *cwnd_* refers to the size of congestion window. It means that the sender of TCP increases the sending rate by about $\alpha \cdot pktSize\_ / RTT$ each *RTT*. RAAR emulates this mechanism at receivers. One of most critical problem is how to estimate *wnd_* at receivers. At receivers, *wnd_* can be estimated as follows,

```
for a packet arrival
    if the packet is the first packet of
    a round
        wnd_=dyWnd_
        dyWnd_=1
else
        dyWnd+=1
```

If there is a packet loss event at the congestion avoidance state, the receiver decreases *rate_* to $\beta \times rate\_$, where β is a constant set in advance. Through our simulations, we discover the reasonable value of β is 0.875. Using equation 1 which is relationship between α and β, we can conclude α is 0.31

B.2 Decision of Packet Loss Event

The Internet is a shared best-effort network with a high level of statistical multiplexing. The observed loss pattern has a near random behavior [5] that is determined by the aggregate traffic pattern. Thus, it is generally hard for an end system to predict or control the loss rate by adjusting the sending rate. The end system can only control the congestion of the network using AIMD adaptation rate mechanisms. However, the only way to attain the network congestion information is to detect the loss event. It takes one round-trip time RTT for end systems to detect and react to congestion. Thus, an end-system only needs to react at most once per RTT as long as it reacts sufficiently. In a RTT, several packet losses are actually caused by the same network congestion. In order to differ from the packet loss, we define a *packet event* as all packet losses appearing during a RTT. Only the packet loss event can show the network congestion correctly. That is to say, in a same RTT only the first packet loss can cause a new packet loss event, while those following packet losses belong to the same packet loss event because they are caused by the same congestion.

It is easy for the receiver to detect packet losses. We can add a timer T with duration of RTT to the RAAR receivers. When a first packet loss appears in a RTT, we can consider a new packet loss event happens, and at the same time, the receiver starts up the timer T. Before the timer expires, we can consider succedent packet losses belong to the same packet loss event and a same congestion causes them. In summary, RAAR end-systems detect the change of the network congestion state by the packet loss event, that is, if there isn't any packet loss event between the previous data packet received successfully and the latest one, the sending rate is increased, whereas, decreased.

B.3 Estimation of RTT And RTO

In RAAR, the round-trip time value RTT and retransmit timeout value RTO are used to determine the length of round and timeout of ACK packets (see Section 3.A). They are measured at receivers.

The way in which RAAR estimates RTT is similar to TCP. The receiver feedbacks an ACK packet to the sender and records the sending time t_0. When the sender receives the ACK packet, it begins a new round immediately. Then the sender sends the first data packet of the new round to the receiver. The receiver records the time t_1 on receiving the data packet, so $t_1 - t_0$ is a sample of RTT. The receiver smoothes the sample of RTT to get a SRTT value using exponentially weighted moving average. The receiver could derive the retransmit timeout values RTO using the usual TCP algorithm:

$$\mathrm{RTO} = \mathrm{SRTT} + 4 \times \mathrm{RTT}_{\mathrm{var}}$$

where $\mathrm{RTT}_{\mathrm{var}}$ is the variance of RTT and SRTT is the smoothed round-trip time. For the other sections of this paper RTT refers to SRTT, otherwise, it is explicitly stated.

B.4 Smoothness Function of Rate

In RAAR, the receiver estimates the sending rate values *rate_*, and reports the sending rate values to the sender. Before the receiver sends ACK packets with the sending rate values, RAAR should smooth the sending rate values. We use the same way as that used in the TFRC protocol [9]. Using an array *r*(*i*) to record the latest *n* historic values of *rate_*, a weighted average rate value *Rate_* is calculated as follows:

$$Rate\_ = \frac{\sum_{i=1}^{n} w(i) \cdot r(i)}{\sum_{i=1}^{n} w(i)}$$

where *w*(*i*) is a weight array, which is defined as: ,

$$w(i) = \begin{cases} 1, & 1 \le i \le n/2, \\ 1\text{-}\dfrac{i - n/2}{n/2 + 1}, & n/2 \le i \le n. \end{cases}$$

The smoothed rate values *Rate_* are reported to the sender by ACK packets. What we should notice is that *Rate_* is the actual sending rate at the sender, while *rate_* is the rate calculated using the GAIMD algorithm at the receiver. *Rate_* cannot replace *rate_* to be used in the GAIMD rate calculation. If so, the TCP-friendly performance of RAAR cannot be guaranteed.

IV. Performance Evaluation

We have tested RAAR extensively in the ns2 simulator [12], and compared it with TFRC protocol by simulations. In this section, we present the major simulation results in detail, which show that RAAR is remarkably fair when competing with TCP flows and its sending rate is reasonably smooth across a wide range of network conditions.

A. Simulation Configurations

For measuring the steady performance of the RAAR protocol, we consider the simple well-know single bottleneck simulation scenario illustrated in Fig. 2. The access links are sufficiently provisioned to ensure that any packet drops/delays due to congestion occur only at the bottleneck bandwidth.

In Fig. 2, R1 and R2 are two routers, and the link between them is the bottleneck. All access links have higher bandwidth and shorter delay than the bottleneck. In all our following simulations, the bandwidth of the access links is 100Mbps, and their delays are random values uniformly distributed between 0 and 20 milliseconds. The bandwidth of the bottleneck is shared by m RAAR (or TFRC), n TCP and k ON-OFF UDP flows. In order to compare fairly, the size of all kinds of packets, including ON-OFF UDP, TCP, RAAR, and TFRC packets, is the same value illustrated in Table 1.

We only test the steady performance of RAAR in this paper. In all following simulations, TCP flows refer to FTP sessions with infinite amount of data. In order to lessen the resonation between sources and reduces the duration of the initial transition phase, all flows are started at uniformly distributed random times. If no special specifications, the simulation parameters are set as ones in Table 1. The throughput for each flow is measured using the number of delivered packets during the last two thirds of the simulation time to ignore transient startup behavior.

B. TCP-Friendliness

The simulation results in this section give us confidence that RAAR is TCP-friendly when competing with TCP traffic of different flavors in the same bottleneck. The bandwidth of the bottleneck is shared by *n* TCP and *n* RAAR (or TFRC) flows. We vary the number of flows in Fig. 3 and 4, and vary the link rate in Fig. 3. The length of simulation time is 600 seconds. These figures show that the mean throughput over last 400 seconds of simulation. We normalize the throughput of RAAR (or TFRC) and TCP in Fig. 3 and 4, so that a value of one would be a fair share of the link bandwidth. Fig. 5 shows the intra-fairness using the value equality fairness.

Fig. 3 illustrates the fairness of RAAR when competing with Sack TCP traffic in both Drop-tail and RED queues. They illustrate that RAAR and TCP co-exist fairly across a wide range of network conditions, i.e. different link rates, drop rates (or number of flows) and queuing algorithms.

We have evaluated a representative curve in Fig. 3 in detail. Fig. 4 shows the 15Mbps data points from Fig. 3. Fig. 4a shows the simulation results that RAAR flows compete with the same number of Sack TCP flows in bottleneck, and its Y axes refers to the normalized throughput of the flows while X axes is the total number of the flows in the bottleneck. The results from RAAR and Reno TCP simulation are summarized in Fig. 4c. In order to compare RAAR with TFRC, we re-conduct the simulations of Fig. 4a by the way that RAAR is replaced by TFRC. Their results are summarized in Fig. 4b.

In Fig. 4a and 4c, we exploited the difference among various TCP flavors to access the impact on RAAR flows. The various TCP flavors have an impact on the TCP-friendliness

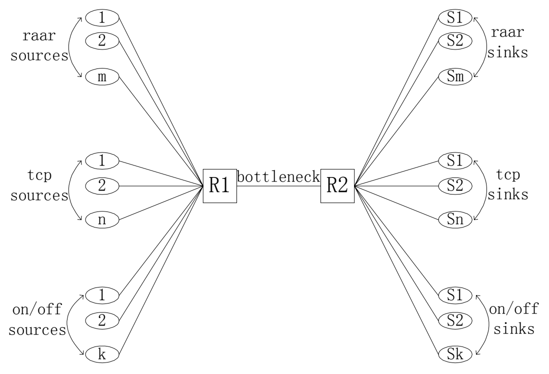
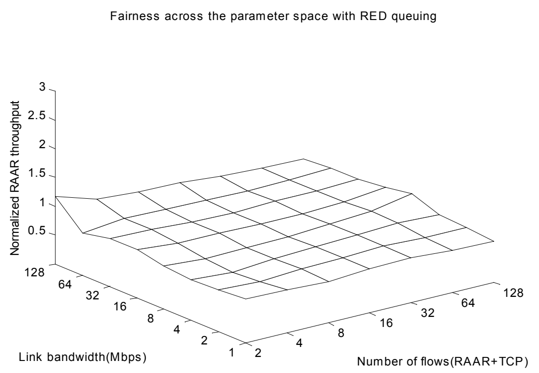


Figure 2. The simulation network topology



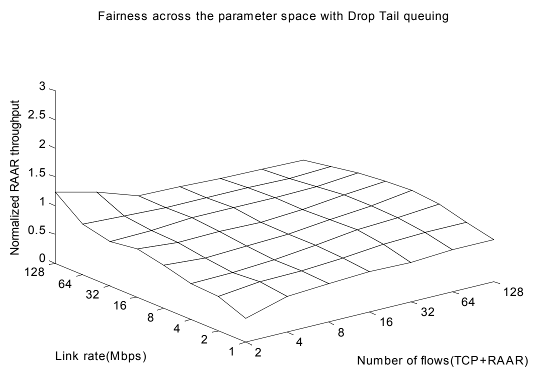


Figure 3. RAAR flow sending rate while co-existing with Sack TCP

of RAAR. Those figures reveal RAAR co-exiting with Reno TCP is more TCP-friendly than with Sack TCP. Because Sack TCP is or will be the most popular TCP flavor and we would like to limit the impact of the TCP's performance problem and focus on the interaction between RAAR and TCP traffic. Therefore, we choose Sack TCP as ideal representative for TCP flows. For the rest of this paper, whenever we refer to TCP, we mean Sack TCP otherwise it is explicitly stated.

Fig. 4a and 4b reveal the comparative results between RAAR and TFRC. The simulations parameters in Fig. 4b are the same as ones in [9], and we utilize the TFRC simulation codes carried in ns2.1b8 [12]. When the number of competing flows becomes more, the performance of TCP-friendliness of RAAR excels that of TFRC. The throughput of TFRC is higher than that of TCP due to the TCP throughput equation that can only predict the upper limit of TCP throughput. Thus, we can find TFRC can occupy more bandwidth on competing with TCP. From the result of Fig. 4a, we consider the TCP-friendliness of RAAR is acceptable.

TABLE I.THE SIMULATION CONFIGURATION PARAMETERS

Packet Size	1000 Byte	TCP/TFRC Overhead	0.002
ACK Size	40 Byte	Mean ON Time	1 s
Bottleneck Delay	50 ms	Mean OFF Time	2 s
Bottleneck Buffers	150 pkts	Rate during ON Time	500 Kbps
TCP Maximum Window	10000 pkts	Shape of ON-OFF	1.5
TCP Tick	100 ms	RAAR betaAIMD_	0.875

Fig. 5 is the value $F_P^{equality}$ of different protocol flows in the above simulations. The value of $F_P^{equality}$, called *equality fairness*, is define as follows:

$$F_P^{equality} = (\sum_{f \in P} R_f)^2 \Big/ (|P| \times \sum_{f \in P} R_f^2)$$

where R_f refers to the average throughput of flow *f*, and |*P*| the number of flows that utilize the protocol *P*

In order to perform comparative studies of TFRC and RAAR, we calculated the values $F_P^{equality}$ of the protocols in the above simulations, and the results are summarized in Fig. 5. It clearly shows that: the intra-protocol fairness of RAAR is approximately equal to that of TFRC when the number of competing flows is few; however, RAAR is better than TFRC when the number of competing flows becomes more. Because the bandwidth of bottleneck is fixed as 15Mbps, the more these are flows in the bottleneck, the higher packet loss rates. Thus, the performance of TFRC is distinctly deteriorated when the packet loss rate of the bottleneck is high (i.e., when there are many flows in the bottleneck). In [24], its simulation results also reveal the same performance problem of TFRC. However, RAAR can perform fairness when the packet loss rate is high.

C. *Performance with Long-Duration Background Traffic*

In this section, we primarily want to test two performances of RAAR. First, we wish to compare the average sending rates (or throughput) of a TCP flow with a RAAR flow experiencing similar network conditions. Second, we would like to compare the smoothness of those sending rate. As a TCP-friendly protocol to transmit multimedia traffic in the future, we would like for RAAR flows to achieve the same average throughput as TCP flows, and yet have less variability. The timescales at which the sending rates are measured affects the values of these measures. In the simulation of this section, we measure the *equivalence ratio* and *Coefficient of Variation* (CoV) [9] of RAAR, TFRC and TCP flows at various timescales.

To compare the sending rates of two flows at a given timescale, the equivalence at time t is defined as follows:

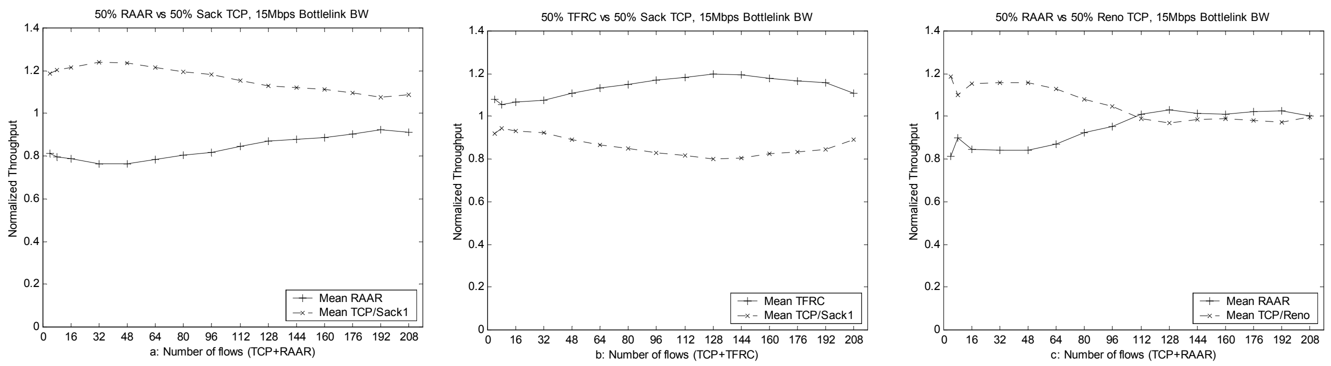


Figure 4. The throughput of TCP and RAAR/TFRC (a: RAAR and TCP/Sack1; b: TFRC and TCP/Sack1; c: RAAR and TCP/Reno)

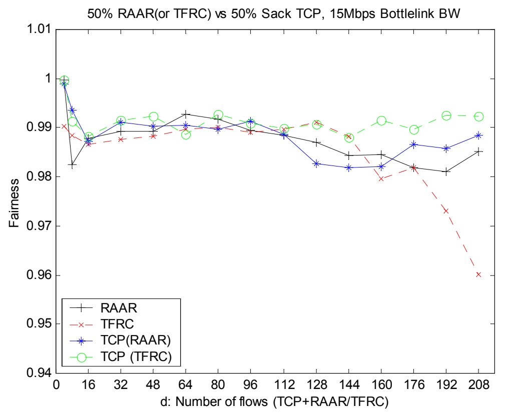


Figure 5. The equality fairness of RAAR/TFRC and TCP

$$e_{\delta,a,b}(t) = \min\left(\frac{R_{a,\delta}(t)}{R_{b,\delta}(t)}, \frac{R_{b,\delta}(t)}{R_{a,\delta}(t)}\right),$$
$$R_{a,\delta}(t) > 0, or, R_{b,\delta}(t) > 0$$

where $R_{f,\delta}(t)$ refers to the sending rate of a given data flow f at time t, measured at a timescale δ. Thus, The average values of the time series, $\{e_{\delta,a,b}(t_0+i*\delta)\}^{n}_{i=0}$, is called the *equivalence ratio* of both flows at a timescale δ. The closer it is to one, the more "equivalent" both flows are.

The coefficient of variation (CoVf, δ) of a given data flow f at a timescale δ, is define as follows [22]:

$$CoV_{f,\delta} = \frac{\sqrt{\frac{1}{(T-t_0)/\delta}\sum_{i=1}^{(T-t_0)/\delta}(R_{f,\delta}(t_0+\delta\cdot i)-R_f)^2}}{R_f}$$

where R_f is average sending rate of a given flow f. The $CoV_{f,\delta}$ can be used as a measure of variability of the sending rate of the flow f at a timescale δ. A lower value $CoV_{f,\ \delta}$ implies a smoother flow.

Fig. 6 and 7 reveal the simulation results from a scenario with a bottleneck of 15Mbps, and 100 packets buffer. The bottleneck queue runs RED queue with *gentle* true, a *minthresh* of 10 and a *maxthresh* of 50. There are 16 RAAR or TFRC protocol flows competing with 16 TCP flows in the bottleneck. The simulation duration is 600 seconds, and the results are from the last 400 seconds of the simulations. The flows are started at random times, uniformly distributed between 0 and 10 seconds. The other simulation parameters can be found in Table 1. The values CoV in those figures are the average values of 16 same protocol flows. The equivalence ratio values in the figures are the average values of a flow and one of the 15 other different protocols flows (or the 15 other same protocol flows). The timescales used in our measurement is 0.2, 0.4, 0.6, 0.8, 1.0, 2.0, 4.0, 6.0, 8.0, 10, 15, and 20 seconds. In the condition, the packet loss rate at the bottleneck is about 0.1%.

Fig. 6 reveals the fairness and the smoothness of RAAR competing with TCP as a function of the timescales of measurement. Curves are shown in Fig. 6a for the mean equivalence ratio between pairs of TCP flows, between pairs of RAAR flows, and between pairs flows of different types. The equivalence ratio of RAAR pairs is bigger than 0.76 and varies little over a broad range of timescales. The equivalence ratio of RAAR and TCP is between 0.6 and 0.8. Thus, we can conclude that the intra-protocol fairness of RAAR protocol excels that of TCP on abroad range of timescales, and the inter-protocol fairness of RAAR and TCP is acceptable. In Fig. 6b, the CoV of RAAR is all less than 0.16 and its variation is very small on a broad range of timescales, while the CoV of TCP depends on the timescales greatly. Especially, the sending rate of RAAR is smoother than that of TCP over a broad range of timescales.

In the same simulation condition, we have compared the fairness and smoothness of RAAR with those of TFRC, that is, we conduct the simulations using 16 RAAR and 16 TCP flows, then, repeat the simulations using 16 TFRC and 16 TCP flows. The results of those simulations are summarized in Fig. 7. The two upper curves in Fig. 7a reveal the intra-protocol fairness of RAAR is equivalent to that of TFRC. The two nether curves show that the inter-protocol fairness of RAAR and TCP excels that of TFRC and TCP. From Fig. 7b, we can find that RAAR flows are smoother than TCP over a broad range of timescales. All comparative simulations in this section utilize the same simulation parameters as ones in [9] and our simulation results of TFRC are close to those in [9]. Thus, we

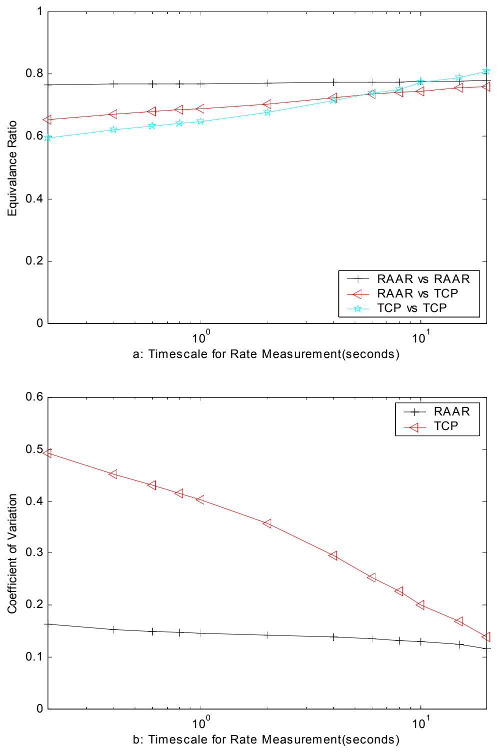


Figure 6: RAAR and TCP's fairness and smoothness in all kinds of timescales (a: Equivalence Ratio; b: Coefficient of Variation.)

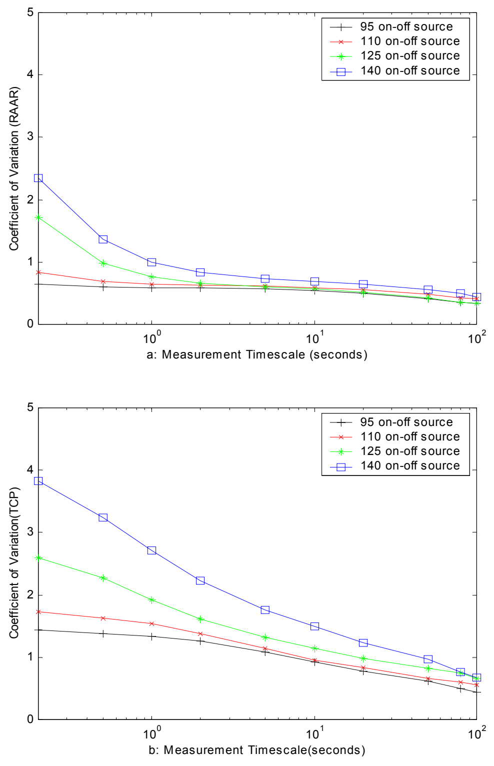


Figure 8. The CoV of RAAR and TCP with ON-OFF as background traffic (a: the CoV of RAAR; b: the CoV of TCP)

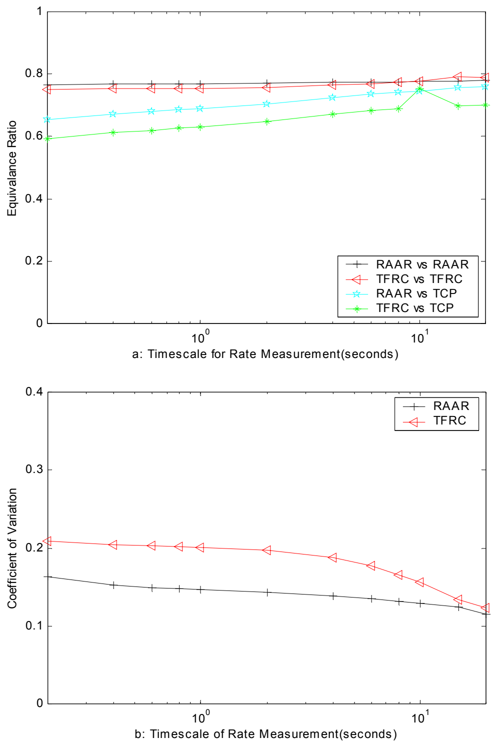


Figure 7. Comparative figure of RAAR and TFRC (a: Equivalence Ratio; b: Coefficient of Variation.)

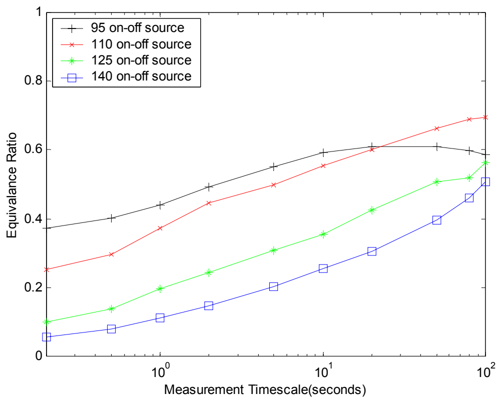


Figure 9. Equivalence ratio of RAAR and TCP with ON-OFF as background traffic

confirm our scripts of the simulations are correct.

From these graphs in this section, we conclude that: 1) on a broad range of timescales, RAAR flows with long duration can share bandwidth fairly with TCP competing flows, and have better rate smoothness; 2) the both performances of RAAR excel TFRC.

D. Performance with Self-Similar Flows as Background Traffic

In this section, we have evaluated the performances of RAAR using a more realistic source model as the Internet

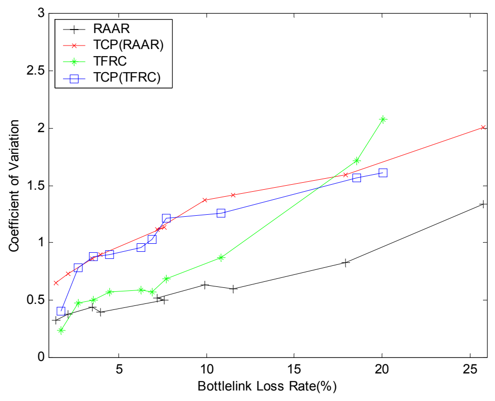


Figure 10. CoV comparison of RAAR and TFRC with ON-OFF as background traffic, and with different packet loss

background traffic. People have found the Internet traffic tends to be self-similar in nature [13]. The self-similar traffic may be created using several ON/OFF UDP sources whose ON/OFF times are drawn from heavy-tailed distribution such as the Pareto distribution [23]. Fig. 8 - 10 present results of simulations in such background. The parameters of ON-OFF UDP data source can be found in Table 1. The simulation duration is 5000 second; the bottleneck queue runs RED with a total buffer of 200 packets, and the other parameters are the same ones as in Table 1 and as in the previous simulations. To test the performance when RAAR and TCP coexisting, we monitor respectively a long-duration RAAR connection and a long-duration TCP connection whose background traffic is self-similar one created using different numbers of ON-OFF UDP flow. The smoothness and fairness of RAAR and TCP are shows in Fig. 8 and 9. In Fig. 10, we utilize variant numbers ON-OFF UDP traffic to generate different packet loss rate in the bottleneck to compare the smoothness of RAAR with that of TFRC and TCP.

With the ON-OFF background traffic, the packet loss rate at the bottleneck is between 10% and 40% in Fig. 8 and 9. At the bottleneck with 95 ON-OFF data sources, we can see that the equivalence ratio of RAAR and TCP sessions is between 0.4 and 0.6 and the CoV of RAAR is between 0.3 and 0.6 over a broad range of timescales. Thus, the results of fairness and smoothness are close to the results in Fig. 6. The two performances of RAAR connections are deteriorated at higher loss rates, such as with 140 ON-OFF data sources (38.25% loss rate). However, on long timescales, even at such high loss rate, the fairness of RAAR competing with TCP is acceptable. From the results in Fig. 8, we can conclude that RAAR is smoother than TCP on a broad range of timescales and at any loss rates.

The Fig. 10 shows that RAAR is smoother than TFRC at a broad range of loss rates. Especially, when the loss rate is very high, the smoothness of TFRC becomes deteriorated, while RAAR can keep up the smoothness very well. The simulation duration of Fig. 10 is 2000 second, we adjust the number of ON-OFF data sources in the bottleneck to create those variant loss rates of the bottleneck, and make a long-duration RAAR (or TFRC) and a long-duration TCP to compete such bottleneck.

The simulation results above present: the fairness and the smoothness of RAAR are acceptable at a broad range of loss rates; the smoothness of RAAR excels that of TFRC when the loss rates of bottleneck are very high.

V. CONLUSION AND FUTURE WORK

In this paper, we have proposed a novel TCP-friendly approach to flow control called Rate Adaptation At Receivers (RAAR) for unicast streaming and it can be upgrade to multicast. We have reported preliminary simulations on verifying performance of the protocol. The simulation results show that we achieve the design goals.

From the simulation results of comparing RAAR with TFRC, we found that both protocols possess desirable performances of fairness and smoothness when their flows compete with TCP flows. However, the both performances of RAAR are better than TFRC, especially, when the packet loss rate is very high. These performance problems of TFRC have been reported in [16], [24]. We suspect that this might be due to inaccuracy in TCP equation itself. In fact, it is difficult to model TCP throughput, so that it's not easy to address the performance problems of the formula-based TCP-friendly protocols. On the other hand, the implementation of RAAR is simpler than that of TFRC.

Currently, we have only simualted long-lived TCP, RAAR and TFRC flows, and only studied their steady performance. Because there are more short-lived flows in current Internet. Although we have conducted simulations with ON-OFF UDP background traffic, the background traffic is not enough accurate to model realistic network traffic. We plan to implement the RAAR algorithm and conducted extensive expeiments to explore the performance of RAAR in Internet.

Lastly, we will develop a multicast version of RAAR. In the unicast version of RAAR, we don't acknowlegde each data packet, which can avoid effectly the feedback implosion. The receivers are charge of almost all workload, which alliviates the sender's burden. Those characteristic are appropriate for multicast.

REFERENCES

[1] C. Albuquerque, B. Vickers and T. Suda. "An End-to-End Source-Adaptive Multi-Layered Multicast (SAMM) Algorithm," in *Proc. 9th International Packet Video Workshop*, New York, April 1999. Also published as UCI-ICS Technical Report 98-31, University of California, Irvine, USA, September 1998.

[2] D. Bansal and H. Balakrishnan. "Binomial Congestion Control Algorithms," in *Proc. of IEEE INFOCOM2001*, Anchorage, AK, April 2001.

[3] B. Braden, D. Clark, J. Crowcroft, etc. "Recommendations on Queue Management and Congestion Avoidance in the Internet," IETF RFC2309, April 1998.

[4] S. Blake et al. "An Architecture for Differentiated Services," RFC 2475 December 1998.

[5] J. Bolot "Characterizing End-to-End Packet Delay and Loss in the Internet," in *Proc. of IEEE INFOCOM1993*, September 1993.

[6] D. Chiu and R. Jain. "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks," *Journal of Computer Networks and ISDN*, Vol. 17, No. 1, June 1989, pp. 1-14.

[7] S. Cen, C. Pu, and J. Walpole. "Flow and Congestion Control for Internet Media Streaming Applications," in *Proc. of Multimedia Computing and networking 1998*, January 1998.

[8] S. Floyd and K. Fall. "Router Mechanisms to Support End-to-end Congestion Control," LBL Technical Report, February 1997.

[9] S. Floyd, M. Handley, J. Padhye, and J. Widmer. "Equation-Based Congestion Control for Unicast Applications," in *Proc. of ACM SIGCOMM2000*, August 2000.

[10] S. Jacobs and A. Eleftheriadis. "Providing Video Services over Networks without Quality of Service Guarantees," in *Proc. of RTMW1996*, Sophia Antipolis, France, October 1996.

[11] J. Mahdavi and S. Floyd. "TCP-friendly unicast rate-based flow control," Technical note sent to the end2end-interest mailing list, January 1997.

[12] NS-2 Network Simulator. http://www.isi.edu/nsnam/ns/index.html, 2001.

[13] K. Park, G. Kim, M. Crovella. "On the relationship between file sizes, transport protocols, and self-similar network traffic," in *Proc. of IEEE International Conference on Network Protocols*, pages 171-180, 1996.

[14] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. "A Model Based TCP-friendly Rate Control Protocol," in *Proc. of NOSSDAV1999*, 1999.

[15] 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. of IEEE INFOCOMM1999*.

[16] I. Rhee, V. Ozdemir, Y. Yi. "TEAR: TCP Emulation at Receivers - Flow Control for Multimedia Streaming." http://citeseer.nj.nec.com/rhee00tear.html.

[17] D. Sisalem, H. Schulzrinne. "The Loss-Delay Adjustment Algorithm: A TCP-friendly Adaptation Scheme," *Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, UK, July 8-10, 1998.

[18] W. R. Stevens. "TCP/IP Illustrated, Volume 2." Addison-Welsley, Reading, MA, November 1994.

[19] Thierry Turletti, Sacha Fosse Parisis, and Jean-Chrysostome Bolot. "Experiments with a Layered Transmission Scheme over the Internet," Research Report No 3296, INRIA

[20] B. Vickers, C. Albuquerque and T. Suda. "Source-adaptive Multi-layered Multicast Algorithms for Real-time Video Distribution," *IEEE/ACM Transactions on Networking*, December 2000. Also published as *UCI-ICS Technical Report 99-45*, University of California, Irvine, USA, October 1999.

[21] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification." IETF RFC2205, September 1997.

[22] J. Widmer. "Equation-Based Congestion Control," Diploma Thesis, February 2000. http://www.icsi.berkeley.edu/~widmer/tfrc/thesis.ps.gz

[23] W. Willinger, V. Paxson and M. S. Taqqu. "Self-Similary and Heavy Tails: Structural Modeling of Network Traffic," Preprint 1996. Appears on pages 27-53 in the book: "A Practical Guide To Heavy Tails: Statistical Techniques and Applications." Robert Adler, Raise Feldman and Murad S. Taqqu., editors. Birkhauser, Boston, 1998.

[24] Y. R. Yang, M. S. Kim, S. S. Lam. "Transient Behaviors of TCP-friendly Congestion Control Protocols," Networking Research Laboratory, Department of Computer Sciences, The University of Texas at Austin. Technical Report TR-2000-14, July 2000.

[25] Y. Richard Yang and Simon S. Lam. "General AIMD Congestion Control," Technical Report TR-2000-09, May 9, 2000. Networking Research Laboratory, Department of Computer Sciences, The University of Texas at Austin. An abbreviated version to appear in *Proceedings ICNP2000*, Osaka, Japan, November 2000.