Supporting Non-Adaptable Multimedia Flows by a TCP-Friendly Transport Protocol

Shengyuan Jan and Wanjiun Liao
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
Email: wjliao@ ntu.edu.tw

Abstract This paper studies TCP-friendly congestion control to support non-adaptable multimedia flows over the Internet. Existing TCP-friendly congestion control mechanisms assume that flow senders can adjust their sending speeds at a fine granularity to match the requests from the congestion control protocols. In reality, the companion CODEC at the sender guides the source sending rates. To date, the supportable rates of any CODEC are "discrete," not "continuous." Therefore, from the prospective of the transport layer protocols, these data flows are non-adaptable. This renders existing congestion control mechanisms inadequate for real environments. In this paper, we propose a TCP-friendly congestion control protocol called Adaptive MultiRate Transport Protocol (AMRTP) to support non-adaptable multimedia flows. We also conduct simulations to evaluate the performance of AMRTP. The results show that AMRTP is TCP friendly, and also has superior performance as compared to existing work.

Keywords: TCP-friendly, non-adaptable flows,

I. INTRODUCTION

In this paper, we study TCP-friendly congestion control to support multimedia applications over the Internet. Existing TCP-friendly congestion control mechanisms [1-4] assume that flow senders can adjust their sending speeds at a fine granularity to match the requests from the congestion control protocols. However, this assumption is impractical for most multimedia applications, such as video transmission. In reality, the companion CODEC at the sender guides the source sending rates. To date, the supportable rates of any CODEC are "discrete" rather than "continuous." In other words, the sending rates can only be adjusted according to a predefined set of coding rates provided by the CODEC, not *any* rate requested by the lower layer protocols. Therefore, from the prospective of the transport layer protocols, these data flows are non-adaptable. In [5], the authors propose a probabilistic way to perform congestion control for single-rate, non-adaptable flows. However, emerging multimedia communications should be of multiple rates, rather than a fixed rate. This renders existing work inadequate, and is the primary motivation of our work.

In this paper, we propose a new TCP-friendly congestion control protocol called "Adaptive MultiRate Transport Protocol (AMRTP)," which supports a discrete set of data sending rates (i.e., non-adaptive multimedia flows) at the flow sender. The major components of AMRTP are an adaptive-AIMD mechanism and a probabilistic decision making module. Note that AMRTP is friendly to TCP Reno plus SACK, considering it is widely used globally.

The rest of the paper is organized as follows. In Section II, a new TCP-friendly congestion control mechanism called Adaptive Multi-rate Transport Protocol (AMRTP) is proposed. In Section III, simulation results are provided to evaluate the performance of AMRTP. Finally, the paper is concluded in Section IV.

II. ADAPTIVE MULTI-RATED TRANSPORT PROTOCOL (AMRTP)

A. Protocol Description

AMRTP is a TCP-friendly transport protocol designed for multimedia sources which generate data packets at several predefined rates, say, $R_1, R_2, \cdots, R_n$, and $R_1 < R_2 < \cdots < R_n$. The source starts a transmission with base rate R_1. The sending rate is then adjusted according to the probed network bandwidth. If there is still bandwidth available, the sending rate increases; if the network is congested, the sending rate decreases.

To be TCP-friendly in sending rate adjustment, a virtual congestion window (denoted as *vcwnd*) similar to *cwnd* in TCP is maintained at each AMRTP sender. The sending rate is then determined as $R_i \le vcwnd / RTT$, where R_i is the nearest rate in $\{ R_1, R_2, \cdots, R_n \}$ satisfying this condition. The growth of *vcwnd* follows AIMD as *cwnd* in TCP, but without the saw-tooth like behavior suffered by conventional AIMD-based congestion control, thanks to the addition of an *adaptive-window-controller based on the congestion history*. To support a discrete set of sending rates at multimedia sources, a probabilistic rate decision mechanism is used. The actual sending rate of each flow is determined by the decision mechanism. The decision is made in such a way as to make the aggregate flow

0-7803-8603-5/04/$20.00 ©2004 IEEE

throughput TCP-friendly, even though individual flows do not behave in a TCP-friendly way.

The operation of the AMRTP receiver is very simple. The receiver replies with one ACK packet per data packet, instead of using cumulative ACKs as in TCP. Based on the received ACK for each packet, the round trip time (RTT) is measured. Note that packet transmission in AMRTP is governed by the determined sending rate, instead of the ACK clock as in TCP.

AMRTP has a slow start phase and a congestion avoidance phase as in TCP, but does not have loss recovery because some packet losses are usually tolerable for many multimedia applications.

B. Slow Start

Like TCP, the slow start phase in AMRTP is used to fast probe the available bandwidth in the network. In this phase, the sending rate is "approximately" doubled every RTT. It is "approximated" because the "actual" sending rate is determined by the sending rate supported by the upper layer application/CODEC.

Let $R_1, R_2, \cdots, R_n$ be the sending rates supported by an application source, and $R_1 < R_2 < \cdots < R_n$. Initially, the sender starts a transmission with base rate R_1 (packets per sec). The sender waits for an ACK from the receiver for the first packet transmitted. If an ACK packet is received before timeout, the round trip time is estimated; otherwise, the sender's timer is doubled as in TCP (i.e., exponential backoff). Based on the estimated RTT, the virtual window (i.e., *vcwnd*) is initialized as $vcwnd = 2 \times R_1 \times RTT$. Thereafter, the *vcwnd* is then doubled per RTT, i.e., $vcwnd = 2 \times vcwnd$. The sending rate will be upgraded to the nearest R_i satisfying

$$R_i \le \frac{vcwnd}{RTT}.$$

Once the ratio of *vcwnd* to *RTT* exceeds the maximum sending rate R_n, the *vcwnd* size will stop growing.

The slow start phase stops when a packet is lost or *vcwnd* reaches the slow start threshold *ssthresh* (set as in TCP). At this time, the congestion avoidance phase is entered.

C. Congestion Avoidance

In the congestion avoidance phase, *vcwnd* grows in such a way as to prevent the network from congestion. For AMRTP, the growth of *vcwnd* is governed by the proposed adaptive-AIMD(α,β), i.e., the value of (α,β) can be configured dynamically according to network conditions, instead of being fixed as in existing work.

In TCP, the instantaneous sending rate is approximately equal to the ratio of congestion window to round-trip-time. Similarly, an AMRTP sender also regards the ratio of *vcwnd* to *RTT* as the instantaneous TCP-friendly sending rate. If the sender cannot find a CODEC matching this rate, the actual sending rate will be determined probabilistically. The probabilistic decision is made so that the aggregate flow rate is TCP-friendly.

1) Adaptive AIMD Congestion Control

In G-AIMD (α, β), a larger α and β gives better transient response to network changes, but may lead to larger sending rate variations. On the contrary, a smaller α and β has smoother sending rate, but suffers from sluggish response to network dynamics. It is not easy to choose a fixed and appropriate value of (α,β) to fit all network conditions for G-AIMD. To avoid the negative effect caused by the wrong choice of (α,β) for G-AIMD, the (α, β) in the proposed adaptive-AIMD is tuned adaptively according to network conditions.

The basic idea behind adaptive-AIMD is described as follows. If the network condition is steady (i.e., the amount of traffic sharing the same bottleneck does not change and the bottleneck bandwidth remains the same), the TCP-friendly sending rate will stay unchanged over time. From this observation, we can lower both the increasing speed (i.e., α) and the decreasing amount (i.e., β) of *vcwnd* when congestion occurs. As such, the rate variation due to the saw-tooth-like problem is reduced while the TCP-friendly requirement is still satisfied. However, when the network condition is changing, we should raise the values of α and β.

To achieve this goal, the AMRTP sender maintains a history window (denoted as *history_wnd*) for the last M occurrences of network congestion. Each element in the history window is the value of *vcwnd* at a congestion instance in the past. Let $vcwnd_k$ denote the latest value of *vcwnd* in congestion. Thus, *history_wnd* = $\{vcwnd_{k-M+1}, vcwnd_{k-M+2}, \ldots, vcwnd_k\}$. Every time a congestion signal is received at the sender, the average value of *vcwnd* (denoted as $\overline{vcwnd}$) is calculated, i.e.,

$$\overline{vcwnd} = \frac{1}{M}\sum_{i=0}^{M-1} vcwnd_{k-i}.$$

If $\overline{vcwnd}$ is close to $vcwnd_k$, i.e., $\left|\overline{vcwnd} - vcwnd_k\right| < \varepsilon$, the network is inferred as being static. The decrease parameters β will then be halved and α will be adjusted according to $\alpha = 3\beta/(2-\beta)$. On the contrary, if $\left|\overline{vcwnd} - vcwnd_k\right| \ge \varepsilon$, the network condition is changing and (α, β) is reset to (1,1/2), so as to respond to the change as quickly as in TCP.

Set	pkts/sec	8	32	80	120	160	200	240	320	400	480
1	bps	64K	256K	640K	960K	1.28M	1.6M	1.92M	2.56M	3.2M	3.84M
Set	pkts/sec	8	16	64	128	176	192	256	320	384	512
2	bps	64K	128K	512K	1024K	1.4M	1.54M	2.05M	2.56M	3.072M	4.096M
H.261									MPEG-2		

Table I. Two sets of sending rates

2) Probabilistic Decision for Actual Sending Rates

The ratio of *vcwnd* to the round trip time, called virtual rate R_v, is the TCP-friendly sending rate. Since this virtual rate may not match any of the supported data rates, the actual sending rate is chosen probabilistically to make the aggregate flow throughput TCP-friendly.

The probabilistic decision mechanism works as follows. The virtual sending rate $R_v(t)$ is defined as $R_v(t) = vcwnd(t)/\mathrm{RTT}$, which is a function of the time t. Each time the virtual sending rate of a sender in the congestion avoidance phase reaches R_i, a uniform random variable X_i is generated from the interval (0,1). During the period of using rate R_i, the "excess rate" function $R_E(t)$ defined as $R_E(t) = \Delta_{R(i)} - R_v(t)$ is measured, where $\Delta_{R(i)} = R_{i+1} - R_i$. Again, $R_E(t)$ is a function of *vcwnd* and hence is a function of time. Fig. 2 illustrates the relationship among $\Delta_{R(i)}$, R_i, R_{i+1}, and $R_E(t)$. Every time the *vcwnd* increases, the condition

$$X_i \geq \frac{R_E(t)}{\Delta_{R(i)}}$$

is tested. If the condition holds, the sending rate is increased from R_i to R_{i+1}. Once the sending rate is increased, the sender will stop this measurement until the value of $R_v(t)$ reaches R_{i+1}. When $R_v(t)$ reaches R_{i+1}, another uniform random variable (i.e., X_{i+1}) is generated. This process repeats until the next congestion occurs, at this time the sending rate is decreased using the same process as in rate increase. Note that once the virtual sending rate $R_v(t)$ exceeds R_n, the sender will stop increasing *vcwnd* because the upper layer application cannot generate data any faster. The sender will not change its *vcwnd* until the next congestion event occurs.

III. PERFORMANCE EVALUATION

In our simulation, N traffic flows share the same bottleneck link with bandwidth x Mbps and propagation delay 40 ms, as shown in Fig. 1. All access links to the center routers have a propagation delay of 5ms and a link speed of 100Mbps.

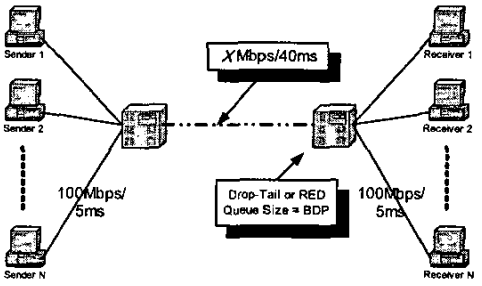

Figure 1. Network topology

In this simulation, no physical errors are assumed during packet transmission. All losses are caused by congestion. The processing delays at end hosts and routers are not taken into account. The maximum queue size (in packets) of the bottleneck router is set to the bandwidth-delay product of the link. The TCP version used is TCP-SACK, implemented as *agent TCP/Sack1* in the ns2 [6] simulator. Each data packet is of size 1000 bytes and the ACK packet is of 40 bytes. The sending rates used in this simulation are summarized in Table I, which also lists their corresponding CODEC types.

We study the TCP-friendliness of AMRTP. Both the intra-protocol fairness (i.e., only considering AMRTP flows), and the inter-protocol fairness between AMRTP and TCP are investigated. The following two performance metrics are defined:

(1) The fairness index f among n flows is defined as $f = \frac{(\sum_{i=1}^{n} R_i)^2}{n(\sum_{i=1}^{n} R_i^2)}$, where R_i is the average sending rate of flow i.

(2) The TCP-friendly ratio is defined as the ratio of aggregate throughput of TCP to that of AMRTP.

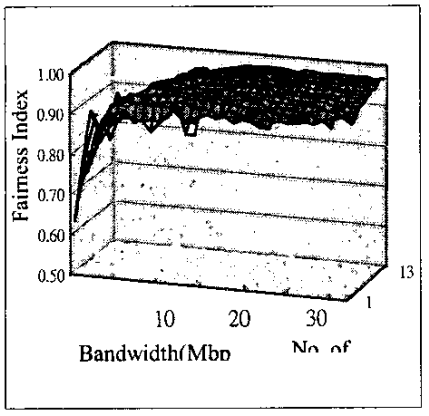

Figure 2. Intra-protocol fairness

We first study intra-protocol fairness. In this simulation, the bottleneck bandwidth B (in Mbps) varies from 0.5Mbps to 16Mbps, and the total number of flows N varies from 2 to 16. Among the N flows, N/2 flows are based on set 1 sending rate and the others use set 2. The flows are randomly selected from the two sets of sending rates in Table I. The start time of each flow is randomly taken from time [0, 0.5] sec.

Fig. 2 plots the fairness index, varying the bottleneck bandwidth and the number of flows. It shows that the fairness indices of AMRTP flows are above 0.99 in the steady state.

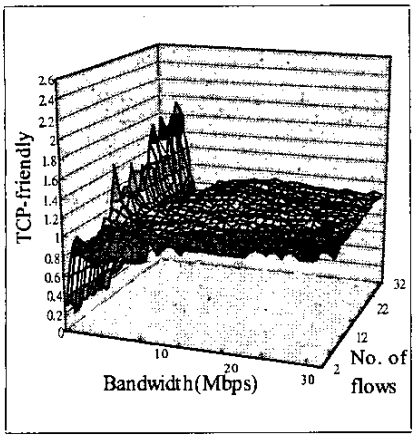

Figure 2. Inter-protocol friendliness

We then study inter-protocol friendliness. In this simulation, the bottleneck bandwidth varies from 500 Kbps to 32 Mbps and the number of flows is set to 3, 6, ..., 30. Among them, one third is TCP, one third is AMRTP using sending rate set 1 and the rest are AMRTP using set 2.

Fig. 3 shows the TCP-friendliness of AMRTP. The TCP-friendly ratio mostly falls between 0.9 and 1.1. When the fair share bandwidth (B/N, i.e., Bandwidth over Number of flows) is small, both TCP and AMRTP flows timeout very often. But since AMRTP works more conservatively and is based a discrete sending rate, it will obtain less bandwidth than TCP in this case (i.e., B/N is small). Note that there is a spike when the bandwidth is around 4 Mbps and the number of flows exceeds 15. The reason is that since AMRTP is not based on the "cumulative-ACK" mechanism, it cannot detect retransmission timeouts as TCP SACK. Thus, the ratio is mostly greater than 1.5 in this region.

IV. CONCLUSION

In this paper, we have proposed a TCP-friendly congestion control protocol called Adaptive MultiRate Transport Protocol (AMRTP) to support non-adaptable, multimedia flows. In AMRTP, two major components are used: adaptive-AIMD and a probabilistic decision maker. Based on the two components, the actual sending rate of each flow is determined in such a way as to make the aggregate flow rate TCP friendly. We have also provided a simple proof of our argument, and conducted simulations to evaluate the performance of AMRTP. The results show that AMRTP is TCP-friendly, and has superior performance as compared to existing work.

ACKNOWLEDGEMENT

This paper is supported by the National Science Council, Taiwan, under Grant Number NSC 93-2213-E-002-001.

REFERENCES

[1] Y. R. Yang and S. S. Lam, "General AIMD Congestion Control, " IEEE ICNP 2000.

[2] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithm," IEEE INFOCOM 2001.

[3] S. Jin, L. Guo, I. Matta, and A. Bestavros, "TCP-Friendly SIMD Congestion Control and Its Convergence Behavior," IEEE ICNP 2001.

[4] S. Floyd, M. Handley, J. Padhye and J. Widmer, "Equation-Based Congestion Control for Unicast Applications," ACM SIGCOMM 2000.

[5] J. Widmer, M. Mauve and J. P. Damm, "Probabilistic Congestion Control for Non-Adaptable Flows," ACM NOSDAV '02.

[6] "ns-2 Network Simulator," available at http://www.isi.edu/nsnam/ns/.