A Streaming-Protocol Retransmission Scheme without Client-Server Clock Synchronization

Soung C. Liew and Patrick C. K. Wu

***Abstract*—This letter considers an Internet streaming protocol (SP) for the transport of real-time continuous data stream. SP makes use of measured network delay to optimally set retransmission time-outs and scale reliability. The sender streams out packets in a periodic manner, and a NACK is issued by the receiver if it does not receive an expected packet by a certain time. Although it may appear at first glance that the sender's and receiver's clocks must be synchronized, we prove that that the SP would work even if the clocks were not synchronized, a situation not uncommon in the Internet.**

***Index Terms*—Clock synchronization, real-time, reliability, retransmission, streaming protocol.**

I. Introduction

ON THE Internet, continuous data streams, such as those from video or data sources, are often transmitted using a streaming protocol. The streaming protocol paces the transmission of consecutive chunks of data so that the receiver would not be overwhelmed with sudden bursts of data. Typically, the data are sent out in a periodic manner.

Many streaming protocols in use today are built on top of UDP (User Datagram Protocol) [1], which is inherently unreliable because it makes no attempt to retransmit lost packets. One could use a TCP-like (Transmission Control Protocol) [2] protocol which aims for 100% reliability by retransmitting the data until a correct copy is received. However, this poses a problem for real-time applications because reliable data, if arriving late at the receiver, are not useful either. The bandwidth used to retransmit these packets could have been better used to speed up the transmission of the subsequent packets. An interesting issue is how to design a protocol to achieve scalable reliability for real-time applications.

This letter considers one such streaming protocol which we shall refer to as SP. In SP, an elastic buffer is employed at the receiver to store a certain amount of data before the beginning of presentation. So long as lost data is not needed for presentation yet, the receiver can request for its retransmission. There is a delay between the instant at which the data is generated at the sender and the instant at which the data is presented at the receiver. Within this delay budget, the receiver may request for retransmission multiple number of times if necessary. Data arriving after at this delay threshold will simply be treated as being lost. In general, larger delay budget implies higher possible number of retransmissions, hence higher level of reliability.

In SP, the sender streams out packets in a periodic manner. When a packet loss is detected by the receiver, a NACK control packet is issued. We use NACK's rather than ACK's because generally much fewer control packets will be generated, especially if the packet-loss probability is not too high.

To effect the first retransmission, the receiver must estimate the single-trip delay time (STT) from the sender. Based on the knowledge of when a packet is sent out and estimates of STT and its deviation, the receiver can set a time-out instant, the expiry of which will trigger off the first NACK. If the first retransmission fails, the receiver must then request for further retransmissions. For these subsequent retransmissions, the receiver must instead make use of the estimated round-trip time (RTT) statistics to set the proper time-out value.

Although not necessary, SP can be built within the framework of the real-time transport protocol (RTP) [3] and RTP control protocol (RTCP). RTP provides a framework for the manipulation of data with real-time characteristics and it adopts the principle of application level framing (ALF) proposed by Clark and Tennenhouse. RTCP provides information on the quality of data distribution such as the data loss rate, delay jitter, etc.

II. SP Retransmission Scheme

Let us consider in more detail the NACK mechanism for the first transmission of a packet. It appears at first glance that it is necessary for the clocks of the sender and receiver to be synchronized for proper time-out operation. On the Internet, most likely the receiver and the sender clocks are not sychronized. The receiver using its own clock as a reference may expect the sender to send out packet i at time $\tilde{T}_i$ where in fact the actual transmission time is T_i. The difference between $\tilde{T}_i$ and T_i may become larger and larger for unsynchronized clocks. We show that as far as setting the time-out instant is concerned, our proposed approach would not give rise to any problem even if the two clocks were not synchronized, making it readily deployable on the Internet.

Suppose that the server sends out packets in a continuous fashion at the rate of one packet every R seconds. Let us say the receiver initially sets its clock to 0 to correspond to the instant at which it thinks the server sends out packet 0, the very first packet. With respect to the receiver's clock, the receiver

Manuscript received July 21, 1998. The associate editor coordinating the review of this letter and approving it for publication was Prof. C. Douligeris. This work was supported by a Competitive Earmarked Research Grant of the Hong Kong Research Grant Council under CUHK 4368/99E.

The authors are with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong.

Publisher Item Identifier S 1089-7798(99)06553-9.

therefore thinks the server sends out packet i at time

$$\tilde{T}_i = iR; \qquad i \geq 0 \tag{1}$$

Because of clock asynchronism, the actual transmission time of packet i with respect to the receiver's clock is

$$T_i = i(R-\delta) + c; \qquad i \geq 0 \tag{2}$$

where c is the initial offset between the two clocks and δ is rate at which the sender's clock is faster than the receiver's clock. There is no known method to synchronize the two clocks (i.e., derive δ and c) between the sender and receiver if the delay between the sender and the receiver is random and time varying [4]. Such is the case for the Internet.

Let A_i to be the arrival time of packet i as measured by the receiver. Then,

$$A_i = T_i + d_i = i(R-\delta) + c + d_i; \qquad i \geq 0 \tag{3}$$

where d_i is the delay from the sender to the receiver.

In order to compute the time-out instant for packet $(i+1)$, the receiver needs to estimate its arrival time. Let $\hat{A}_{i+1}$ be the estimate. To compute $\hat{A}_{i+1}$, we first need to estimate the single-trip delay time from the sender to the receiver. A smoothing equation similar to that used in the estimation of the round-trip delay time in TCP can be used. Let S_i be the smoothed single-trip delay time based on the perceived delay of packet $i, \tilde{d}_i$. It is given by

$$\begin{aligned} S_i &= \lambda_s S_{i-1} + (1-\lambda_s)\tilde{d}_i; \quad \text{for } i \geq 1 \\ S_0 &= \tilde{d}_0 \end{aligned} \tag{4}$$

where

$$\tilde{d}_i = A_i - \tilde{T}_i = A_i - iR = c + d_i - i\delta; \qquad i \geq 0 \tag{5}$$

and λ_s is a constant value between 0 and 1. Solving the linear difference equation governing S_i, we have

$$\begin{aligned} S_i = {} & c - \left(i - \sum_{n=1}^{i} \lambda_s^n\right)\delta + \lambda_s^i d_0 \\ & + (1-\lambda_s)\sum_{n=1}^{i} \lambda_s^{i-n}\, d_n; \qquad i \geq 1 \end{aligned} \tag{6}$$

The receiver can estimate the arrival time of packet $(i+1)$ by adding S_i to the instant at which the receiver thinks the sender sends out packet $(i+1)$

$$\hat{A}_{i+1} = \tilde{T}_{i+1} + S_i = (i+1)R + S_i \tag{7}$$

The error of the estimation is given by

$$\begin{aligned} e_{i+1} &= \hat{A}_{i+1} - A_{i+1} \\ &= S_i + (i+1)\delta - c - d_{i+1} \\ &= \sum_{n=0}^{i} \lambda_s^n \delta + \lambda_s^i d_0 + (1-\lambda_s)\sum_{n=1}^{i} \lambda_s^{i-n}\, d_n - d_{i+1}. \end{aligned} \tag{8}$$

The first term of (8) is due to clock asynchronism, and the other terms are due to delay and are present regardless of whether the clocks are synchronized or not. If the delay d_i is constant for all i, then these other terms sum to zero.

The absolute value of the first term is smaller than but approaches $|\delta/(1-\lambda_s)|$ as $i \to \infty$. In particular, the term does not diverge to infinity even though the clock difference between the sender and the receiver grows indefinitely for nonzero δ. Recall that δ is the clock drift during the time interval between the sending of two successive packets and should therefore be very small. For instance, if the time that elapses between the sending of two successive packets is in the millisecond regime, and the clock frequency is accurate up to one part in 10^3 (a very modest assumption given the present technology), δ is in the microsecond range. As for $(1-\lambda_s)$, it is typically set to be about 0.1 in practice. We can therefore conclude that clock asynchronism will not be a significant factor in our estimation of the arrival time of a packet.

In SP, the time-out instant for packet $(i+1)$ is set to be $\hat{A}_{i+1}$ plus a safety margin. The safety margin should be related to $e_i, e_{i-1}, e_{1-2}, \cdots$ since they indicate the likely deviation of the actual arrival time of packet $(i+1)$ from its estimate. We can use a smoothed value of $|e_i|, D_i$, to set the time-out instant. Specifically,

$$D_i = \lambda_e D_{i-1} + (1-\lambda_e)|e_i| \tag{9}$$

for some constant λ_e value between zero and one. The time-out instant for packet $(i+1)$ is

$$TO_{i+1} = \hat{A}_{i+1} + \beta D_i \tag{10}$$

for some constant $\beta > 1$.

In the above derivation, we have assumed that all packets are received. In reality, packets can be dropped. As in TCP [2], only the statistics of the received packets should be taken into account when updating the values of S_i and D_i. Specifically, the following set of equations is used to set the various parameters:

$$\begin{aligned} S_{\text{new}} &= \lambda_s S_{\text{old}} + (1-\lambda_s)\tilde{d}_{\text{new}} \\ D_{\text{new}} &= \lambda_e D_{\text{old}} + (1-\lambda_e)|\hat{A}_{\text{new}} - A_{\text{new}}| \\ \hat{A}_i &= iR + S_{\text{new}} \\ TO_i &= \hat{A}_i + \beta D_{\text{new}} \end{aligned} \tag{11}$$

where the index *new* refers to parameters based on the last received packet and the index *old* refers to parameters based on the second last received packet.

For illustration, Fig. 1 shows the trace of an example where there is a difference in the rates of the sender's and receiver's clocks. In the figure, $R = 6$ ms, $\delta = -3$ ms, $c = 0$ ms, the single-trip time is normally distributed with mean 100 ms and standard deviation 20 ms. The assumed packet loss probablity is 0.1. We set $\lambda_s = 0.875$ in our tracking algorihtm. ET_i in the figure corresponds to $\tilde{T}_i$ in the text, and EA_i is $\hat{A}_i$. As shown, even with rather drastic clock rate difference—the sender clock's rate is 33% slower than the receiver clock's rate—and the high packet loss probabiliy of 0.1, the expected arrival time $\hat{A}_i$ can track the actual arrival time A_i without them drifting apart.

The above discussion concerns the time-out mechanism for the first transmission of a packet. If the packet is lost in the

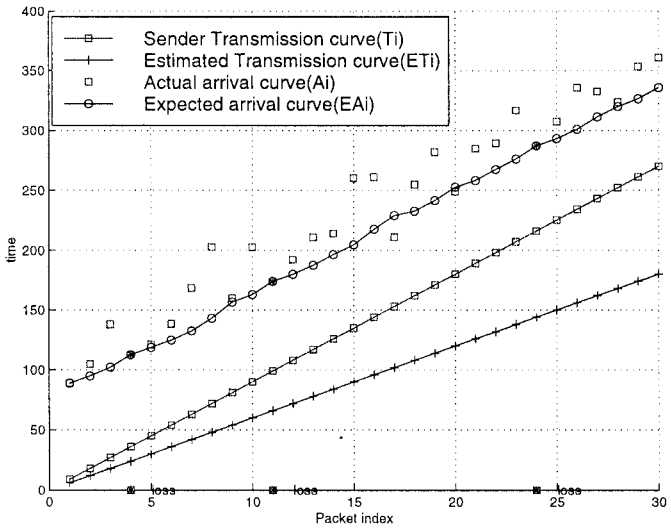

Fig. 1. Transmission and arrival curves.

network and time-out occurs, a NACK will be issued and there will be a retransmission. The retransmitted packets themselves may get lost again. For retransmissions, the time-out instant should be based on the round-trip delay rather than the single-trip delay because it is initiated by the receiver. That is, when the receiver sends a NACK, it should set a time-out period equal to an estimate of the round-trip time plus a safety margin. Clock asynchronism between the sender and receiver is not an issued because, unlike the single-trip delay, round-trip delay is an entity that can be measured using only the receiver's clock.

Let r_{new} be the measured round-trip time for the latest NACK with a response returning from the sender. Denote the previous estimate of the round-trip time by R_{old}. Then, as in the estimate of the single-trip time, we can use the following smoothing equation to obtain a new round-trip time estimate R_{new}:

$$R_{\text{new}} = \lambda_R R_{\text{old}} + (1 - \lambda_R) r_{\text{new}} \tag{12}$$

where λ_R is a constant value between zero and one.

Let us denote by D^R the deviation estimate of the round-trip time. The following smoothing update equation can be used:

$$D^R_{\text{new}} = \lambda_{D^R} D^R_{\text{old}} + (1 - \lambda_{D^R}) |r_{\text{new}} - RTT_{\text{new}}| \tag{13}$$

where λ_{D^R} is a constant value between zero and one.

The time out instant TO^R_i for NACK i can be set as follows:

$$TO^R_i = T^R_i + R_{\text{new}} + \beta^R D^R_{\text{new}} \tag{(14)}$$

where T^R_i is the time at which NACK i is issued and β^R is a constant larger than one.

If the packet-loss probability is very small, then very few NACK's are issued. Since the above estimate of the round-trip time makes use of the responses to NACK's, it may not very accurate under such circumstances. However, one may argue that if the system is already very reliable, the accuracy of the estimate is not so vital since only very rarely will a second NACK be needed to receive a correct copy of a packet. If the system is unreliable and many NACK's are issued, the estimate scales accordingly and becomes accurate automatically.

Alternatively, if SP is built within the framework of RTP and RTCP, the round-trip-time can be obtained from the RTCP [5]. RTCP makes use of a separate channel to measure its value using probe packets. In this case, SP periodically checks the round-trip-time value given by RTCP and derives a smoothed average using the above equations.

III. Summary

We have described a streaming protocol for real-time Internet applications. Hosts connected to the Internet often have unsynchronized clocks. We have shown that the retransmission scheme of the protocol works even when the sender's and receiver's clocks are not synchronized. Currently, implementation is underway to test audio transmission using the protocol. In addition, we are investigating the use of the protocol for multicast applications.

References

[1] J. Postel, "RFC-768: user datagram protocol," *Request for Comments*, Aug. 1980.
[2] J. Postel, "RFC-793: Transmission control protocol," *Request for Comments*, Sept. 1981.
[3] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "Draft-ietf-avt-rtp-new-00.ps: RTP: A transport protocol for real-time applications," *Internet Draft*, Jan. 1996.
[4] V. Paxson, "Measurements and analysis of end-to-end Internet dynamics," Ph.D. dissertation, Univ. California, Berkeley, Apr. 1997.
[5] P. Karn, C. Partridge, "Improving round-trip time estimates in reliable transport protocols," *Computer Communicators Rev.*, vol. 17, no. 5, pp. 2–7, Aug. 1987.