Packet Scheduling in R2CP and Priority-Based R2CP for Video Streaming: A comparative Study

Kamalakshi N[#1], H Naganna[*2]

[#1,2]*Dept of Computer Science & Engg Sapthagiri College of Engg Bangalore,India*

[1] kamalnags@yahoo.co.in, [2]naganna_h@hotmail.com

***Abstract*— R2CP(Radial Reception Control Protocol) does not distinguish the data packets which are critical to the quality of video streaming, so there is a great risk that the critical data packets may be dropped when network is in congestion. This paper describes P-R2CP (Priority-based R2CP) to effectively decrease the loss ratio of critical data packets in multipoint-to-point video streaming.**

Keywords*— congestion control, streaming, buffer management, flow control, *TCP- friendly

I. INTRODUCTION

Operations via the conventional TCP are sender-driven, and receivers only place the received packets into the buffer in sequence. In real-time video streaming, the TCP retransmission mechanism will rather reduce the transmission performance. RCP is designed to use most of the TCP transmission mechanisms, including a congestion window-based control mechanism, such as the binomial streaming-friendly congestion control [4]. The main difference is that RCP reverses the TCP transmission direction. The streaming operations will be receiver-driven, and the sender will only respond to receiver's requests. When the receiver detects that some data packets cannot arrive in time, it will directly drop the packets and ignore the retransmission mechanism to avoid the reduction of the transmission performance. In P2P network, its topology is not fixed all the time. Like in the ad hoc mode of wireless transmissions, each peer node has a different online and offline time to-peer networks.

R2CP is a receiver-driven, multi state transport protocol that supports multipoint-to-point connections The R2CP destination (receiver) maintains multiple states, each of them corresponds to the single state maintained by individual sources (senders) in the connection

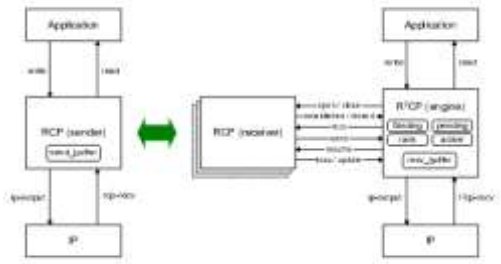

Figure 1. R²CP Architecture

R2CP is built atop a single state, point-to-point transport protocol called RCP (Reception Control Protocol) RCP is a TCP clone in its general behavior, including the use of the same window-based congestion control mechanism. However, RCP transposes the intelligence of TCP from the sender to the receiver such that the RCP receiver is primarily in charge of the congestion control and reliability. The RCP receiver drives the progression of the connection, while the RCP sender merely responds to the instructions sent by the receiver. It is shown in [14] that RCP is indeed TCP- friendly. Multiple RCP pipes in an R2CP connection are coordinated by the R2CP engine at the receiver. The R2CP engine is responsible for buffer management, flow control, and the reliable delivery of data to the application, while individual RCPs implement congestion.

Multiple RCP pipes in an R2CP connection are coordinated by the R2CP engine at the receiver. The R2CP engine is responsible for buffer management, flow control, and the reliable delivery of data to the application, while individual RCPs implement congestion control. Note that although RCP by itself is a reliable protocol like TCP, R2CP allows data recovery to occur along the RCP pipe different from the one data was sent. This is achieved in R2CP through dynamic binding of the application data (to be requested) and the RCP packets using the binding data structure Effectively, individual RCPs control how much data to request from each sender, while the R2CP engine control which data to request from each sender.

II. PACKET SCHEDULING IN R2CP

While individual RCP pipes provide transmission slots for data requests from the sources, a key task that needs to be performed by the R2CP engine is to decide

which data to request for each transmission slot; that is, to schedule requests of application data. The objective is to minimize the number of packet losses at the receiver due to either buffer overflow or dead- line expiry. We observe that an optimal schedule (that minimizes the number of packet losses) for streaming pocket with non-decreasing deadlines to U buffer limited receiver is one that ensures in sequence arrival of packets at the receiver. Intuitively, in-sequence delivery minimizes the chances of buffer overflow at the receiver. Moreover, since the deadlines associated with packets of increasing sequence numbers are non-decreasing, in-sequence delivery also minimizes the chances of packets missing the deadlines.

To minimize out of order arrivals, the R2CP engine needs information regarding the band widths and latencies of individual pipes. Since congestion control is a process of bandwidth estimation, changes in bandwidth and latency are directly reflected through the changes in the Size of the congestion window. Hence, the progression of the congestion window in RCP provides an effective way for the R2CP engine to track the available bandwidth along each pipe. The R2CP engine uses the arrivals of data to ides an effective way for the R2CP engine to track the available bandwidth along each pipe. The R2CP engine clocks the transmissions of new requests, and schedules a request along a path only when the concerned RCP pipe has space in its congestion window for requests. Since the request (REQ) in RCP has the dual role of the acknowledgment (ACK) in TCP for congestion control [2], the use of such a fine-grained packet scheduling allows R2CP to closely track the bandwidth fluc- tuitions without incurred extra overheads compared to TCP.

While one side way to schedule a request is to assign the next global sequence number to the new request, such first-come-first-served (FCFS) scheduling will cause out-of-order arrivals due to mismatched latencies along dif- ferent paths. The assignment instead should be hazed on the potential order (rank) of data packets arrivals. The R2CP engine maintains a rank data structure for finding the rank of the new request. Specifically, whenever the R'CP engine sends out a request for sequence number I through pipe j: an entry is added to the rank data structure with a timestamp of S = Ti + RTT,, whir T, is the time when the request is sent, and RTT, is the round-trip time of pipe j. The timestamp reflects the time when a new request wills he issued in response to the arrival of the requested data. When the R2CP engine receives the .send() call from pipe k at time T, it locates the rank r of the request as = The R2CP engine uses the arrivals of data to clock the transmissions of new requests, and schedules a request along a path only when the concerned RCP pipe has space in its congestion window for requests. where N is the total number of entries in rank, S,, is the timestamp at entry n,p(n) is the ipe that sent out the request at entry n, and by+ is the maximum of zero and the largest integer less than or equal tox. In other words, the R2CP engine finds the number of data segments that will he requested after T (due to arrivals of pending requests) hut arrive before T + RTTk. Once the rank r is identified, the R2CP engine hinds the dh data in line to the new request, and adds an entry for the request in the rank data structure as usual. The entry is deleted when the corresponding data arrives.

III. PRIORITY BASED R2CP

The packet scheduling mechanism of R2CP is able to effectively reduce the amount of data packets that not arrived in order and the total amount of packets lost, when it is applied to video streaming, it does not take into account the importance of video frames (e.g. I-, P-, and B-frame). Therefore, important data packets may be mistakenly dropped, causing the reduction of playback quality. In P-R2CP packet scheduling method, the concept of priority is integrated, so it will be more suitable for multipoint-to-point video streaming**.**

Suppose there are n data packets, and each of which is given a serial number from 1, 2, 3, …n. Suppose there are a total of m RCP connections and each can be denoted by RCP_i , i ∈ { 1, 2, 3…, m }. Suppose that data packet j has to arrive before a deadline j, j ∈ { 1, 2, 3, …, n }. „ Suppose the time stamp of data packet transmitted via RCP_i is $S = T_j + RTT_i$

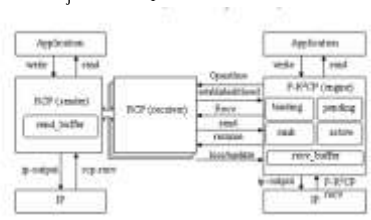

Fig.2 P-R2CP

As shown in Fig.2 P-R2CP engine manages four data structures and the receiver buffer.

1) Binding: It provides the mapping of local serial numbers and global serial numbers. P-R2CP engine at the receiver side will bind global serial numbers and requested data packets, and each RCP connection will manage the local serial number of the protocol. When any RCP connection issues send() to request transmission, the local serial number will be converted by the P-R2CP engine in a packet scheduling process into a global serial number. The sender of each RCP connection will process the receiver's request according to the global serial number. (Note that the global serial number is the serial number of data packets).
2) Pending: It provides the range of serial numbers of data packets pending to be requested, including the serial numbers to be retransmitted and the serial numbers that have not been requested and dropped.

3) Rank: When RCPj requests for data packet with the serial number i, a time stamp $S = T_i + RTT_j$ will be added in rank, where T_i is the requested time of transmission and $_{RTTj}$ is the round trip time of RCP_j. This time stamp reflects the expected time that the requested data packet 'i' arrives at the receiver.
4) Active: When P-R2CP engine receives send () from a RCP connection, and recv_buffer has no more space for more data packets (due to application backlog), it will freeze this RCP connection and include it into active. Later, when there is space in recv_buffer (due to application reading), P-R2CP engine calls for resume() to activate the RCP connection in active.

IV. PRIORITY BASED PACKET SCHEDULING

First of all, all the I-, P-, and B-frames are segmented into data packets of 1024 bytes, and each packet is assigned a serial number. Assume that P-R2CP engine receives send() from RCP_k at time T and the local serial number is l, according to the rank formula proposed in [1], the rank of this request r can be calculated as follows.

$$r = 1 + \sum_{d=1}^{N} \left\lfloor (T + RTT_k - S_d) / RTT_{p(d)} \right\rfloor^{+}$$

where N is the total items in the rank, S_d is the time stamp of item d in rank, p(d) is the RCP connection of item d in rank, and the symbol + means to take the largest number between 0 and largest integer smaller or equal to x. In the next step, P-R2CP engine finds the serial number 'g' of the r-th data packet in pending and verify if $T + RTT_k \leq$ deadline 'g' is true. If true, P-R2CP engine binds 'g' and this request, adds a time stamp $S = T_g + RTT_k$ in rank (Note that $T = T_g$), and removes the serial number 'g' from pending. Otherwise, it will find in pending the smallest serial number 'q' and verify if $T_g + RTT_k \leq$ deadline 'q'. It binds q and this request, adds a time stamp $S = T_q + RTT_k$ in rank (note that $T = T_q$), and removes q from pending. Later, P-R2CP engine adds an item (k, l) in binding where k represents the index of RCP_k and 'l' represents the operating parameters of RCP_k. Moreover, P-R2CP creates a 'reversed' tag in recv_buffer for the arrival of the data packet g or q. Finally, the largest serial number from among the arrived data packets h is also bound with this request, which is then transmitted to the sender of RCP_k. In the above process of searching for the packet with the smallest serial number q which compiles $T + RTT_k \leq$ deadline 'q', it is necessary to search for the largest serial number s that is smaller than t and whose importance is also smaller than that of t, where t is between g and q-1. If 's' is found, it will be removed from pending; or if data packet t is a B-frame packet, t will also be removed from pending first (to drop it in advance). Otherwise, the engine will passively wait for other RCP connections to transmit data packet 't' according to packet scheduling. If t cannot be delivered in time, it will be dropped too. Generally speaking, dropping in advance can shift forwards the RCP connections after s. Based on the assumption that packet scheduling makes data packets arrive at the receiver in sequence (and effectively reduces the number of late data packets), this shifting process can potentially increase the possibility that the important data packet t arrives in time.

V. PROTOCOL OPERATIONS

As illustrated in Fig. 2, if the serial number to be bound with the request is g, when the request data packet arrives at the sender of RCP_k, the sender will retrieve the data packet g from send_buffer and transfer it to the receiver RCP_k. Later, the sender will remove the data packets with a serial number smaller than 'h'. Because the sender only responds to any request of data from the receiver, non-sequential and discontinuous transmission of data packets is likely to happen. After data packet g arrives, P-R2CP engine will put it in the reserved place of recv_buffer and search for the corresponding (k, l) where k represents the index of RCP_k and l represents the operating parameters of RCP_k. It uses recv () to transfer l to RCP_k and Removes the corresponding (k, l) and $T_g + RTT_k$ from binding and rank. Later, RCP_k will update its status and determine whether it can send more requests according to the congestion window. If more requests can be made, send() will be used to request transmission. If data packets have not arrived in sequence three times or any data packet loss is detected [5], RCP connections will use loss() to inform P-R2CP engine and remove the corresponding items in binding and rank to cancel the binding with the serial number of the lost data packet. According to the reliability of RCP, this unbound serial number will be reinserted into pending for retransmission or removal. If RCP connections have an updated RTT, they will use update () to notify P-R2CP engine to update rank. Whenever an RCP connection is established or terminated, P-R2CP engine uses open () and close () to add or delete the connection. If a RCP connection is terminated, all the serial numbers bound with the terminated data packets will be cancelled and reinserted into pending.

VI. CONCLUSION

Our study observed the drawback of R2CP that data packets cannot be dropped by their significance to the video quality and then improved R2CP by P-R2CP to effectively reduce the rate of mistakenly dropping important data packets

REFERENCES

[1] H. Y. Hsieh and R. Sivakumar, "Accelerating Peer-to-peer Networks for Video Streaming Using Multipoint-to-Point

Communication," IEEE Communications Magazine, pp. 111-119, August 2004.

[2] H. Y. Hsieh, K. H. Kim, Y. Zhu, and R. Sivakumar, "A receiver-centric Transport Protocol for Mobile Hosts with Heterogeneous Wireless Interfaces," ACM MOBICOM, San Diego, CA, pp. 363-382, Sept. 2003.

[3] D. Wu, Y. T. Hou, W. Zhu, Y. Q. Zhang, J. M. Peba, "Streaming video over the internet: approaches and directions," IEEE Trans. Circuits Syst. Video Technol., vol. 11, pp. 282-300, Feb. 2001.

[4] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms," IEEE INFOCOM, Anchorage, AK, pp.631-640, Apr. 2001.

[5] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A transport protocol for real-time applications,

[6]] D. Bansal and H. Balakrishnan, "Binomial Congestion Control

[7] J. [Byerr. M. Luby, and M. Mitzenmacher. "Accessing Multiple Mirror Sites in Parallel: Using Tornado,Coder to Speed Up Downloads." Proc. IEEE INFOCOM. New York, NY, Mar. 1999.

[8] H.-Y. Hrieh and R. Sivakumar, "A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-Homed Mobile Hosts." Proc. ACM MOBICOM. Atlanta. GA, Sept. 2002.

[9] Apostolopol et al.. "On Multiple Description Streaming with Content Delivery Networks." Pro'. IEEE INFOCOM. New York. NY. June 2002

[10] T. Nguyen and A. Zakhor. "Distributed Video Streaming over Internet," Pmc. SPIE MMCN, San lose, C4 Jan. 2002.

[11] H.-Y. Hsieh et al.. "A receiver-centric Transport Protocol for Mobile Hosts with Heterogeneous Wireless Interfaces," Prac ACM MOBICOM, San Diego, C4 Sept. 2003.

[12] D. Bansal and H. Balakrirhnan "Binomial Congestion Control Algorithms." Proc. IEEE INFOCOM, Anchorage.AK. AD. 2001.

[13] D.' Ciark and D. Tennenhouse, "Architectural Consideration for a New Generation of PrOtOCOIS," Proc. ACM SIGCOMM. Philadelphia. PA, Sept. 1990

[14] N. Feamiter and H. Balakrishnan. "Packet Loss Recovery for Streamino Video." Proc. Packet Video Wksr)..