A MAX-MIN FAIRNESS CONGESTION CONTROL FOR STREAMING LAYERED VIDEO

Hsu-Feng Hsiao and Jenq-Neng Hwang
Dept. of Electrical Engineering, University of Washington, Seattle WA 98195
{hill, hwang}@ee.washington.edu

ABSTRACT

In a best-effort networking environment, efficient and fair congestion control is highly desired for every traffic flow to share the bandwidth appropriately. This paper proposes a congestion control algorithm for UDP based layered video, whose bandwidth resolution in each layer has been predefined. This proposed congestion control mechanism is an extension of XCP, which is a newly proposed protocol believed to be superior to TCP, especially for high bandwidth-delay product networks. This paper also introduces *reserved packet length* so that the traffic of layered video can share the bandwidth of a network better with the consideration of max-min fairness to other traffics.

1. INTRODUCTION

To overcome the scalability issues of video streaming dissemination over heterogeneously networked receivers, the video-content providers would either prepare several copies of a video, each at different bit-rate, or have the video encoded in layered structure so that the receiver can acquire an appropriate copy based on the network condition and play the incoming video in real time. One common transport protocol to deliver the contents is RTP on top of UDP, which basically provides neither guarantee on the quality of service nor friendliness to competing traffics. Some would adopt RTCP as a control protocol to regulate the data rate [1]. The objective of this paper is to define a congestion control mechanism for layered video to approach fairness to background traffics. The fairness criterion used here is max-min fairness [2], which assigns the largest possible resource to the flow with lowest throughput.

The majority of the internet traffic nowadays is still TCP, which follows an end-to-end protocol and adjusts the congestion window based on the model with factors such as packet loss and round trip time [3]. Several approaches in literature have reported that TCP-based congestion control has fairness problems and has difficulty in utilizing the network resource efficiently in high per-flow bandwidth-delay product networks [4]-[6]. For fairness issues, one example is that even with two receivers sharing the same bottleneck of a network, the steady throughputs are different as long as their round trip times are different.

Many researches [6]-[9] have worked on improvements of TCP congestion control. In the Explicit Congestion Notification (ECN) protocol [7], the routers pass forward the network condition using a single bit in the IP header to notify the presence of congestion. Extended from ECN, routers with Explicit Control Protocol (XCP) [6] explicitly fill in a few bits of the header with some details of network condition to enable senders to quickly and accurately adapt to the assigned fair bandwidth using window-based rate control. With the verification of *ns2* simulations [6], XCP proves to be much superior to TCP in terms of fairness and convergence time of bandwidth variation, both in typical or high-speed (Giga range) networks.

Layered video, on the other hand, behaves quite differently from the window-based TCP (or XCP) flows. Layered video enables a receiver to decode and display a subset (usually the accumulative collection) of layered video it can receive. The resolution of the changes of data-rate is determined by the bandwidth of a video layer and has a property of quantum change. A bigger number of layers generally ensure greater bandwidth resolution. Usually, depending on the coding method, there is a trade-off between the number of layers and the coding efficiency of video. Thus, the sender of layered video can not transmit whatever the assigned data rate the network is telling it to transmit.

In this paper, we examine the aggregated behavior of available bandwidth of a node in XCP. We then extend XCP to include rate-based congestion control and further propose the reserved bandwidth strategy through reserved packet length for layered video to fairly compete the traffic.

The remaining paper is organized as follow. In Section 2, we discuss the congestion control algorithm for rate-based traffics as an extension of XCP originally proposed by [6], and the modified version for the estimation of spare bandwidth. Section 3 proposes reserved packet length for layered video. Section 4 presents simulation results, followed by the concluding remarks in Section 5.

2. SPARE BANDWIDTH ANALYSIS

With more built in intelligence, today's routers can take advantage of some useful collected information to estimate spare bandwidth and thus lead to more reasonable bandwidth allocation for flows passing through them. XCP [6] suggests a stateless congestion control model for window-based rate control. This paper extends their concepts to rate-based congestion control and even to flows which can only have quantum changes on bandwidth such as layered video.

2.1. Aggregate Feedback in the Node

XCP introduces a new congestion header which contains elements such as congestion window ***cwnd***, round trip time ***rtt***, and a feedback field indicating spare bandwidth [6]. The sender fills the first two header elements while routers dynamically update the (per-packet) feedback field. An XCP receiver acts similarly to TCP by sending back acknowledgement (***ack***) packets except it also attaches the feedback in the ***ack*** packets to the sender. The sender updates the congestion window based on the accumulated feedback in the ***ack*** packet and current ***cwnd***. While the throughput of a window-based XCP flow can only be implicitly represented by ***rtt*** and ***cwnd***, the throughput of a rate-based UDP flow can be indicated explicitly in the header.

As shown in Fig. 1, the simplest way to infer the spare bandwidth of a given node would be (C_{out} - R_{in}), where C_{out} is the outgoing-link capacity and R_{in} is the incoming rate. A more sophisticated definition, called aggregate feedback φ, has been given by [6]:

$$\varphi = \alpha \cdot (C_{out} - R_{in}) - \beta \cdot \frac{Q_{\min}}{d}. \tag{2.1}$$

where α and β are constant parameters; the control period d is estimated by the average ***rtt*** of all the flows; Q_{min} is the minimum queue during the last propagation delay.

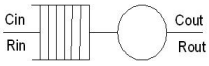

Fig. 1. Node model.

When φ>0, φ is divided equally to all the flows. On the other hand, it is allocated to flows proportionally to their current throughputs when φ< 0. The aggregate feedback information is an aggregated calculation of packets passing through the node. To avoid keeping track of the per-flow status in the router, the router can only insert traffic information in each packet's (per-packet) feedback field when it passes by, that is why the per-packet feedback information needs to be derived from the aggregate feedback φ without the knowledge of flow status. The per-packet feedback is expressed as the difference of positive feedback and negative feedback, i.e., (p_k - n_k) [6]:

$$p_k = \varepsilon_p \cdot \frac{s_k}{r_k}, \text{where } \varepsilon_p = \frac{h+\max(\varphi,0)}{\sum \frac{s_i}{r_i}};$$
$$n_k = \varepsilon_n \cdot s_k, \text{where } \varepsilon_n = \frac{h+\max(-\varphi,0)}{\sum s_i}. \tag{2.2}$$

where s_k is the packet length (available in the IP header) of packet k; r_k is the flow throughput specified either implicitly in TCP/XCP header or explicitly in the header of our layered video packets. Bandwidth shuffling is introduced to avoid convergence stalling at φ=0 [10] and h is the shuffled data rate which is about 10% of the traffic.

Once all the per-packet feedbacks are collected by the sender for rate-based applications, the total change in the throughput of a flow is equal to the sum of the per-packet feedbacks it receives for that flow.

2.2. Modification of Aggregate Feedback Formulation

Equation (2.1) is an important indicator to spare bandwidth. The second term on the right side is required so that the queue can be consumed even at C_{out} = R_{in}. Instead of using Q_{min}, which is more difficult to keep track of, in our formulation we choose to use the queue length Q_{end} at the end of each period, which is the queue amount needed to be consumed in the next duration. This selection of queue length provides more realistic simulation results in our experiments. The aggregate feedback can now be reformulated as:

$$\varphi = \alpha(C_{out} - R_{in}) - \beta \frac{Q_{\text{end}}}{d}. \tag{2.3}$$

3. RESERVED BANDWIDTH FOR LAYERED VIDEO

In the XCP protocol, when the sender receives the per-packet feedback from the ***ack*** packets, it is supposed to update the congestion window right away to reflect its share of spare bandwidth along the end-to-end path [6]. For layered video, however, the sender cannot increase the sending rate before its accumulated per-packet feedback is greater than the data rate of the higher layer to be subscribed. When the feedback is negative, it needs to reduce the sending rate at the resolution of layer bandwidth which is no less than the amount suggested by the feedback.

To consolidate the feedback packets, the receiver can send back only an ***ack*** packet over time duration t_d, which is chosen to be ***rtt*** of current flow. This ***ack*** packet carries the information of the accumulated per-packet feedback $p_{sum\_t_d} = \sum_{\text{over time } t_d} (p_k - n_k)$ and the sender can also update the ***rtt*** value upon receiving this ***ack*** packet. Before the sender acquires the first consolidated ***ack*** packet so as to determine the ***rtt*** value, a pre-defined initial value is used.

Alternatively, the ***rtt*** value from the initial control signal (TCP packets) to set up this streaming video application can be used as the initial ***rtt*** value for the UDP session.

To preserve the spare bandwidth before the sender can actually transmit a new layer to the receiver, we introduce the concept of reserved packet length Δs_k as the additional information placed in the packet header. When routers calculate the per-packet feedback for each incoming packet as suggested in (2.2) and (2.3), the packet length s_k is replaced by $(s_k + \Delta s_k)$. The information Δs_k written in the header shall be always greater than zero. When the calculated Δs_k is less than zero, the sender needs to withdraw a layer (or layers) and readjust the reserved packet length Δs_k, as discussed in Section 3.2.

When the sender receives a new accumulated feedback $p_{sum\_t_d}$ in the ***ack*** packet, the recursive equation to update Δs_k is:

$$\Delta s_k \Leftarrow \Delta s_k + \delta s_k. \tag{3.1}$$

From the routers' point of view, the difference of receiving rate introduced by δs_k should be equal to the accumulated feedback. Specifically,

$$p_{sum\_t_d} = N_k \cdot \delta s_k \Rightarrow \delta s_k = \frac{p_{sum\_t_d}}{N_k}. \tag{3.2}$$

where N_k is the *packet rate* (packet number per second) transmitted to the receiver, which can be derived as follows:

$$N_k = \frac{r_k}{s_k}. \tag{3.3}$$

The resulting receiving rate $\tilde{r}_k$ of flow *k* observed in the routers, taking into account of the reserved packet length Δs_k, is now:

$$\tilde{r}_k = r_k + N_k \cdot \Delta s_k = r_k \cdot (1 + \frac{\Delta s_k}{s_k}). \tag{3.4}$$

With the reserved packet length in the loop, the shared spare bandwidth can be actually preserved by recording Δs_k in the header instead of being taken away by other flows in the next iteration.

For sake of clearer discussion in the later subsection while without loss of generality, we assume every video layer has the same bandwidth r and packet size s. We also assume that the receiver currently has m video layers. It is important to determine when to add a new layer (or layers) and to discard a layer (or layers) with respect to the changes of reserved packet length Δs.

3.1. When $\Delta s \geq 0$

When the reserved bandwidth is enough for at least an extra layer:

$$\frac{mr}{s}(s + \Delta s) \geq (m+1) \cdot r, \tag{3.5}$$

$$\Delta s \geq \frac{s}{m}. \tag{3.6}$$

Thus, when $\Delta s \geq s/m$, the sender can start to transmit i additional layers to this receiver. Once the number of transmitted layers m is indeed increased, the new reserved packet length $\hat{\Delta}s$ should also be updated accordingly to reflect this increase of layers:

$$\frac{mr}{s}(s + \Delta s) = \frac{(m+i)r}{s}(s + \hat{\Delta}s). \tag{3.7}$$

$$\hat{\Delta}s = \frac{m \cdot \Delta s - i \cdot s}{m+i}. \tag{3.8}$$

Since the new $\hat{\Delta}s$ needs to be greater than or equal to zero, we can derive number i as the largest integer satisfying the following equation.

$$\begin{aligned} &\hat{\Delta}s = \frac{m \cdot \Delta s - i \cdot s}{m+i} \geq 0, \\ &i \leq \frac{m\Delta s}{s}. \end{aligned} \tag{3.9}$$

3.2. When $\Delta s < 0$

On the other hand, when $\Delta s < 0$, which indicates that the sender needs to withdraw $i (\leq m-1)$ layers from being sent to this receiver. As a result, the bandwidth would be reduced by $(r \cdot i)$. The reduction of bandwidth $r \cdot i$ could be more than the indicated feedback by the routers. To compensate for this over-reduction, a new reserved packet length $\hat{\Delta}s$ is derived:

$$\frac{mr}{s}(s + \Delta s) = \frac{(m-i)r}{s}(s + \hat{\Delta}s). \tag{3.10}$$

$$\hat{\Delta}s = \frac{m \cdot \Delta s + i \cdot s}{m-i}. \tag{3.11}$$

Since the new $\hat{\Delta}s$ needs to be greater than or equal to zero,

$$\begin{aligned} &\hat{\Delta}s = \frac{m \cdot \Delta s + i \cdot s}{m-i} \geq 0, \\ &i \geq \frac{-m\Delta s}{s}. \end{aligned} \tag{3.12}$$

Hence i would be the smallest integer satisfying (3.12). Thus, congestion control for layered video can be made to be compatible with congestion control in XCP and thus create the max-min fairness behavior.

An alternative way to make use of the preserved bandwidth is to put error recovery code for adjacent frames or packets as an attachment in the current packet, whose length is set to be equal to the reserved packet

lengths as shown in (3.9) and (3.12), to be able to recover some of the packet errors such as packet loss.

4. SIMULATIONS

We use network simulator, ns2 [11], to evaluate the performance of the congestion control on layered video applications. Pseudo packets for streaming layered video are generated at 200 Kbps per layer. Dumb-bell shaped network topology (see Fig. 2) is used with bottleneck occurring at node 2 to node 3 with capacity 2000 Kbps. The queue at each node can store up to 80 packets. Senders are placed at node 0, node 1 and node 7, while receivers are at either node 5 or node 6.

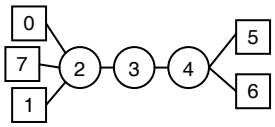

Fig. 2. Network topology.

In Fig. 3, there are three flows starting at different time instances. The solid and dotted lines are the throughputs for FTP-XCP traffics. Dash-dotted line is the real throughput for layered video. There are transient layer changes at around time sec 10, but overall the response time is reasonably fast and fair to other traffics which are confined to similar congestion control algorithm. As also indicated in [6], this congestion control mechanism can avoid built-up queue quite efficiently. Fig. 4 is the corresponding queue length (in terms of packet number) at bottleneck node 2. In Fig. 5, we consider the same topology in addition to an intrusive UDP traffic at 1000 Kbps starting from time sec 7 to sec 15 (UDP throughput is not shown in Fig. 5 for better comparison with Fig. 3). Fig. 6 is the queue length at node 2. This invading traffic doesn't have any congestion control and thus well-controlled traffics are starved; nevertheless they still converge to their static state quickly. How to deal with the ill-mannered traffic is left as future work.

5. CONCLUSION

It has been a popular topic to design a congestion control mechanism for streaming video data to be fair to the competing traffics. In this paper, we generalize the window-based congestion control in XCP [6] to include rate-based layered video traffic. To counter the abrupt changes of staircase-like bandwidth in layered video, reserved packet length is introduced. The results show that layered video with proposed congestion control mechanism can share the bandwidth with other well-behaved traffics quite well.

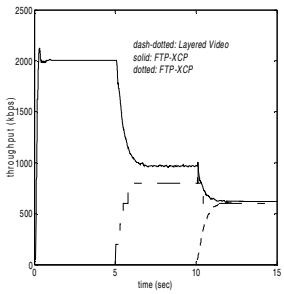

Fig. 3. Flow throughputs.

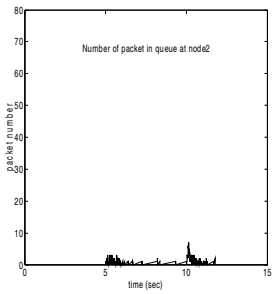

Fig. 4. Queue at node 2.

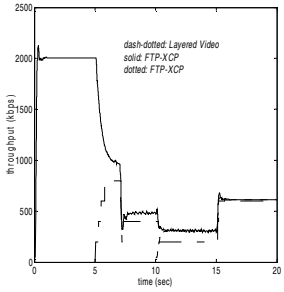

Fig. 5. Flow throughputs.

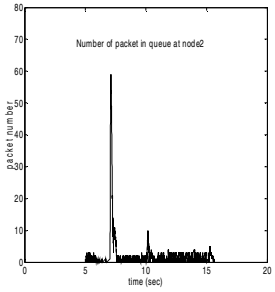

Fig. 6. Queue at node 2.

6. REFERENCES

[1] D. Wu, Y.T. Hou, W. Zhu, Y.Q. Zhang and J.M. Peha, "Streaming Video over the Internet: Approaches and Directions," *IEEE Trans. Circ. and Syst. for Video Tech.*, vol. 11, no. 3, pp. 282-300, March 2001.

[2] D. Bertsekas and R. Gallager, "Data Networks," *Prentice-Hall*, 1987.

[3] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *RFC2001*, Jan 1997.

[4] G. Hasegawa and M. Murata, "Survey on fairness issues in TCP congestion control mechanisms," *IEICE Trans. Commun.*, June 2001.

[5] S. Shalunov, "Gigatcp-tcp on gigabit ethernet," *Internet2/NLANER Joint Tech. Workshop Boulder*, July 2002.

[6] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," *ACM Sigcomm* 2002.

[7] S. Floyd "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, 1994.

[8] K. Ramakrishnan, S. Floyd, D. Black "The Addition of Explicit Congestion Notification (ECN) to IP," *RFC 3168*, Sep. 2001.

[9] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *ACM Trans. On Networking*, Aug. 1993.

[10] D. Katabi, "Decoupling Congestion Control and Bandwidth Allocation Policy with Application to High Bandwidth-Delay Product Networks," *Ph.D. Dissertation*, March 2003.

[11] ns2, http://www.isi.edu/nsnam/ns/.