Routing-based Video Multicast Congestion Control

Jun Peng and Biplab Sikdar

Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
110 8th St., Troy, NY 12180, USA

Abstract. Congestion control is critical for a multicast transport protocol to be deployed and coexist fairly with current unicast transport protocols, such as TCP. We present a new congestion control protocol for video multicast: Routing-based Video Multicast Congestion Control (RVMCC), which combats congestion from a new direction: enriching abstractions of the routing layer. RVMCC overcomes most of the disadvantages of current end-to-end multi-layer video multicast congestion control schemes, such as unstable throughput and unfair sharing of bandwidth with other sessions [9] [10]. These disadvantages are inherent for end-to-end multi-layer video multicast congestion control schemes and extremely hard for them to deal with [10]. RVMCC not only achieves good stability of throughput but also approaches Max-Min fairness closely at bottlenecks. The former is necessary for ensuring the viewing quality of transmitted video, while the latter is necessary for the deployment of multicast in the current Internet. Furthermore, unlike existing network-assisted video multicast congestion control schemes, RVMCC does not require the change of the queuing, scheduling, or forwarding structure of the current Internet. RVMCC can be integrated with minimum assistance from network by enriching the abstractions of the routing layer.

1 Introduction And Previous Work

When a video source needs to transport a piece of video to multiple receivers, it can set up a unicast session with each receiver and send one copy of each packet to each of them at each transmission. A more economic way to accomplish this is to set up one multicast session with all receivers and send a single copy of each packet to the first hop router at each transmission, and then multicast routers on the multicast tree duplicate each packet when the packet reaches forks of the multicast tree. Although the advantages of multicast are obvious, the deployment of multicast lags far behind. One hurdle for the deployment is the lack of a mature multicast congestion control scheme. Congestion must be avoided at any branch of the multicast tree, but at the same time each receiver should receive data at the highest rate affordable to the path leading to it. This goal is hard to achieve because of the heterogeneity of receivers and of links leading to them. For example, if a video multicast source needs to serve three receivers at the same time, and the bandwidth of the access links of the three receivers is 64Kb/s, 128Kb/ and 256Kb/s, respectively, which rate should the video source choose to serve these receivers?

One smart way to solve this heterogeneity problem was first suggested in [1] [13] and elaborated on or reported in [2] [3] [4] [5] [6]. The basic idea is to encode video into several number of layers and multicast each layer to a different multicast group. The receivers then adapt to congestion by adding and dropping layers (i.e. joining and leaving multicast groups). In the last example, video source can be encoded into 4 layers with a rate of 64Kb/s per layer. So the three receivers in our example can subscribe to 1, 2 and 4 layers, respectively, and then every receiver is satisfied. When available bandwidth changes, they just need to add or drop layers to adapt to it. A transport protocol based on this idea was proposed in [7]: Receiver-driven Layered Multicast (RLM). Another similar protocol that shows more potential for inter-session fairness is Receiver driven Layered Congestion control (RLC) [8].

Although these protocols represent an indisputable advance in congestion control for video multicast, they have several inherent shortcomings [9] [10]. Three main ones are slow convergence, inter-session unfairness and unstable layers and throughput. As pointed out in [10], without changing the foundations of these protocols, these problems are hard to solve. One most important foundation of these protocols is the join experiment: adding layers to explore for available bandwidth. Because of the delay of multicast grafting and pruning, failed experiments come with a high price to pay: a period of congestion. But if they are too conservative in adding a layer, bandwidth will be underutilized or grabbed by other sessions. So it is very hard for them to achieve a balance among bandwidth utilization, stability of layers and fairness between sessions. Although the scheme in [11] solves the join-experiment problem partly, it has some serious disadvantages. Dynamic layering is hard, if not impossible, to realize for video source. Routing overhead is heavy because of continuous joining-group and leaving-group actions of receivers. Furthermore, it still has fairness problem with TCP.

With all of these disadvantages of end-to-end video multicast congestion control schemes, some researchers are exploring another direction to deal with this problem: requesting assistance from network by changing queuing, scheduling or forwarding structures of the network [12] [14]. Although the approach in [14] can improve the stability of video quality, it still has problems in achieving fairness with TCP sessions, since it does not touch the foundation of RLM. Although the scheme in [12] could get good results, there are several disadvantages, as pointed out by the author. First, routers need states of downstream neighboring routers. Second, it requires class-based network service. The last is that the scheduling policy is complex.

In this paper we present a new video multicast congestion control protocol: Routing-based Video Multicast Congestion Control (RVMCC), which deals with the video multicast congestion control problem from a new perspective. Instead of changing the queuing, scheduling or forwarding structure of the current Internet, RVMCC enriches the abstractions of the routing layer to deal with congestion. Since routing protocols reside in routers, it is easy for them to obtain the states of the network and the traffic. They also can respond fast if necessary. In addition, since routing decisions directly apply to flows, the control response is quick. So RVMCC converges quickly. Furthermore, RVMCC achieves not only good stability of throughput but also approaches Max-Min fairness closely. This is because RVMCC can get much more knowledge about the states of the network (such as queue status) and about the going-on traffic (such as the number of sessions competing for the bandwidth at a bottleneck) than end-to-end schemes do.

The rest of the paper is organized as follows. We introduce RVMCC in details in Section 2. In Section 3, we show detailed simulation results of the protocol. Summary appears in Section 4.

2 The RVMCC Protocol

Let us consider transmitting a piece of video to some receivers across the Internet with the RVMCC protocol active in routers. The source encodes the video data into several layers and strips them across several multicast groups (we call all layers from the same source a Video Multicast (VM) session). Receivers of the VM session subscribe to some of these layers to receive video data at appropriate rate. Meanwhile, if congestion occurs at a link where VM sessions are passing through, RVMCC starts to observe that link. Specifically, RVMCC samples the traffic to get the knowledge about the number of VM and TCP sessions passing through the link; the rate of each VM session is also estimated. RVMCC decides whether or not to block a layer of a VM session passing through the link according to the total bandwidth that all VM sessions passing through the link are using. If the VM sessions are using unfair share of bandwidth, RVMCC blocks a layer of the VM session that is using the most bandwidth among the VM sessions. At the same time, RVMCC keeps observing the queue status of the link. When the free-bandwidth state of the link is observed, RVMCC channels a layer of the VM session that is using the least bandwidth among the VM sessions. In this process, receivers add and drop layers to cooperate.

Before we discuss the details of the RVMCC protocol below, we give some definitions to facilitate our description. We base our definition on a link that VM and TCP sessions are passing through.

– *C*: the bandwidth of the link
– *M*: the number of VM sessions passing through the link
– *N*: the number of TCP sessions passing through the link
– *B*: the total bandwidth used by all VM sessions passing through the link
– *link-equal-share*: C / (M+N)
– *VM-equal-share*: B / M
– *VM-share-ratio*: VM-equal-share / link-equal-share = (B / M) * ((M+N) / C)
– *VM-total-link-equal-share*: M * link-equal-share = M * C / (M+N)

2.1 Layer Definition and Layer Priority in RVMCC

In a video multicast session, different layers are usually with different significance. The lower the layer, the higher its priority. In RVMCC, this priority information is embedded in the multicast address that a layer of a session uses. The lower the address, the higher its priority. But this rule only applies to layers of the same VM session, not to layers of different VM sessions.

Each time a VM session applies for multicast addresses, it should be assigned a block of continuous addresses. Then the VM session assigns lower addresses to its lower layers and higher addresses to its higher layers. Furthermore, each VM session is assigned a VM session number by the source host. This session number is used to distinguish two different sessions initiated from the same host. The session number needs only to be unique in the source host but not globally unique. The layers with the same source address and VM session number are assigned different priorities according to their multicast addresses when RVMCC needs to block or channel a layer.

2.2 Solving Congestion and Claiming Bandwidth

Instead of using layer-add and layer-drop as in end-to-end video multicast congestion control schemes, RVMCC uses layer-block and layer-channel to solve congestion and to claim bandwidth, respectively. Layer-block is the modification of the multicast routing table to prevent a layer from entering a congested link, while layer-channel is the modification of the routing table to allow a layer to enter a link. In layer-block, routing states such as the source address of the layer, the multicast address of the layer and the link at which the layer is blocked, are saved and are used later for layer-channel. Those routing states can only be deleted by unsubscription actions of receivers.

In fact, for efficient network resource utilization, the router that just blocked a layer should forward the blocking information upstream. So the upstream routers that do not need to forward that layer to other branches except the congested one may also block that layer. In this way, no layer will reach a router where no downstream receiver will receive the layer. When the router that blocked layers just now needs to channel a layer, it also has to send channeling information upstream to graft the multicast branch leading to it. With this mechanism, although bandwidth will be efficiently used in data transmission, a considerable amount of communication and cooperation will be required among routers.

In RVMCC, blocking information is not forwarded upstream. Instead, receivers actively participate in the adjustment of session rate to simplify the network assistance and to relieve routers of excessive communication and processing burden. Every receiver unsubscribes to all but the lowest empty layer of its session. An empty layer is a layer subscribed to by a receiver but receiving no data because of being blocked somewhere in the network. At the same time, when the empty layer is channeled and data arrives at the empty layer, receivers must subscribe to another layer to prepare for a new empty layer, if more layers are available for subscription.

With this simplified mechanism, although one redundant layer may arrive at some upstream routers above a bottleneck, network and processing overhead is considerably lowered. Furthermore, with this mechanism, data will arrive at receivers faster when congestion disappears, since grafting above the bottleneck is not needed anymore. Especially for video multicast, this is a great advantage.

2.3 Achieving Fairness and Stability

Intra-session fairness is successfully achieved in a VM session. The number of layers that a receiver can receive is not decided by its own actions or by other receivers' actions but only by the bottleneck on the path leading to it from the multicast source. This is not true for most current end-to-end multi-layer multicast congestion control schemes, although they adopt different methods trying to achieve this goal.

Inter-session fairness is a tough topic for all transport protocol design. Even for mature unicast transport protocols, such as TCP, fairness is not always achieved between their own sessions. When two TCP sessions sharing the same bottleneck are with much different RTTs, their shares of the bottleneck bandwidth can be far from equal. RVMCC approaches the Max-Min fairness closely and shows the following characteristics in inter-session fairness: (1) With best effort, every session is assigned the link-equal-share bandwidth; (2) With best effort, spare bandwidth from any session is shared by all other sessions; (3) No session is starved for bandwidth. The design details for achieving these goals are described below.

First, RVMCC ensures that VM sessions get the instantaneous VM-total-link-equal-share. When congestion occurs at a link, if the total bandwidth currently used by all VM sessions passing through the link is less than the VM-total-link-equal-share, RVMCC will not block layers. This has some meaning of class-based service. But for variable bit rate (VBR) video sources, this does not ensure the long-run VM-total-link-equal-share because layers may be blocked during their peak-rate period.

Second, when blocking is necessary, the VM session with the highest share of bandwidth among all VM sessions passing through the link is selected to block a layer. On the other hand, when channeling is necessary, the VM session with the lowest share of bandwidth among all VM sessions passing through the link is selected to channel a layer.

Third, when a new VM session arrives and congestion occurs, the link-equal-share is recalculated. If VM sessions are using more bandwidth than the new VM-total-link-equal-share, some layers of them are blocked.

Fourth, when a new TCP session arrives, RVMCC blocks layers for it only if the VM-share-ratio is larger than a threshold or there is no TCP session passing through the link. When the VM-share-ratio is less than the threshold, the new TCP session has to grab bandwidth from other current TCP sessions. But when a new TCP session arrives and the threshold is reached, the VM-total-link-equal-share is recalculated and some bandwidth is released if VM sessions are using more bandwidth than the VM-total-link-equal-share. This procedure can stabilize VM traffic while not starve TCP sessions at any time. Since TCP sessions may come and go frequently, to respond to every new TCP session may cause unstable VM traffic.

Fifth, when a VM session leaves, the remaining VM sessions claim some bandwidth from it to achieve the new VM-total-link-equal-share. In this process, TCP sessions also get some bandwidth. This is necessary for fairness, since a departing VM session usually releases a significant amount of bandwidth.

The last, when a TCP session leaves, RVMCC only claims some bandwidth from it if the VM-share-ratio is less than a threshold. Otherwise, the bandwidth spared by the departing TCP session is claimed by other TCP sessions. The consideration is also for the stability of VM traffic, since TCP sessions may come and go frequently.

2.4 Further Design for Stability

When losses are observed at a link, RVMCC does not block layers instantly if there is no new session, even if the total bandwidth being used by all VM sessions is greater than the VM-total-link-equal-share. Only if the losses persist for some time longer than a threshold, does RVMCC start to block layers. This procedure further stabilizes VM traffic.

Because of the Additive Increase and Multiplicative Decrease (AIMD) flow control scheme implemented with TCP, TCP traffic usually shows saw-tooth like fluctuation and may cause temporary light congestion. So the above procedure can filter out this kind of temporary but possibly frequent light congestion.

One important point is that, unlike end-to-end video multicast congestion control schemes, RVMCC can afford the time to judge temporary congestion because it can solve the congestion quickly if the congestion turns out to be serious. For end-to-end schemes, delayed response to serious congestion comes with a high price to pay.

2.5 Network Cost

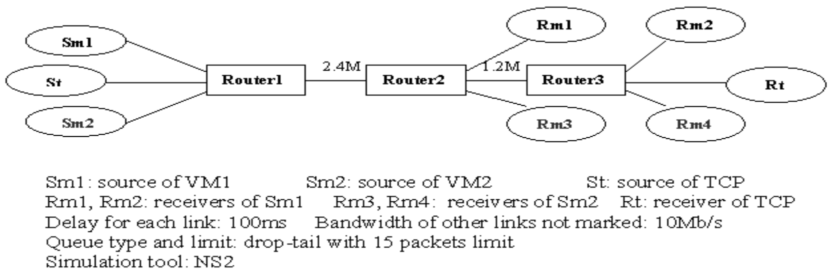
Although the RVMCC protocol does not require any change of the structure of the current Internet, there is some network cost in its operation, mainly the estimation of the number of sessions and the rates of VM sessions passing through a congested link. But because the estimation is only started at a link when congestion occurs at that link during a multicast session, the total cost for a multicast session may not be high. Furthermore, we believe the cost is worthwhile. First, it has not been shown that an end-to-end multi-layer video multicast congestion control scheme can coexist with TCP with some kind of fairness, although research has been going on for some time. Second, other existing network-assisted schemes need considerable change of the current Internet. Finally, the rate estimation of multicast sessions at congested links can prevent the abuse of current Internet resources, intentionally or accidently, by multicast applications. If the cost does become a concern for backbone routers, they can ignore the rate estimation of each multicast session. Instead, they select sessions to block or channel a layer only according to the total number of layers that each session has. Then backbone routers are relieved of excessive burden. This will not affect the scheme significantly because it has been shown that losses in MBone mainly occur on local networks [15].

3 Simulation Results

In this section we present the simulation results. We are mainly interested in observing the convergence, fairness and stability characteristics of RVMCC. Two kinds of VM sources are used: Constant Bit Rate (CBR) sources and Variable Bit Rate (VBR) sources. The VBR sources have exponentially distributed burst-times and idle-times. One source (VBR1) has an average burst time of 200ms and an average idle time of 100ms, while both the average burst time and the average idle time of the other source (VBR2) are 200ms. Each source has 5 layers and the average rate of each layer is 200Kb/s. The throughput shown in our figures is averaged over one second, so it does not represent the instantaneous rate of a session.

3.1 Network Topology and Parameters in Our Simulations

The topology used in our simulations is shown in Fig.1. There are two concatenated bottlenecks in this topology. The first one has a bandwidth of 2.4Mb/s, a capacity of 12 layers, while the second one has a bandwidth of 1.2Mb/s, a capacity of 6 layers. Receivers Rm1 and Rm2 are with the VM session 1. Rm1 is under the first bottleneck, while Rm2 is under the second bottleneck. VM session 2 also has two receivers, Rm3 and Rm4. Rm3 is under the first bottleneck, while Rm4 is under the second bottleneck. Rt is the receiver of the TCP session and located under the second bottleneck. So all three sessions will compete for bandwidth at both bottlenecks if all of them are alive. But Rm1 and Rm3 are only affected by the first bottleneck, while Rm2, Rm4 and Rt are under the influence of both bottlenecks.



Sm1: source of VM1 Sm2: source of VM2 St: source of TCP
Rm1, Rm2: receivers of Sm1 Rm3, Rm4: receivers of Sm2 Rt: receiver of TCP
Delay for each link: 100ms Bandwidth of other links not marked: 10Mb/s
Queue type and limit: drop-tail with 15 packets limit
Simulation tool: NS2

Fig. 1. Simulation Topology

3.2 Simulation Scenarios

Scenario One: Two Competing VM Sessions In this scenario, the interaction between two VM sessions is tested. The first VM session starts at time 0 and stops at the 1200th second, while the second VM session starts at the 100th second and stops at the 1100th second. The results are shown in Fig.2, Fig.3 and Fig.4

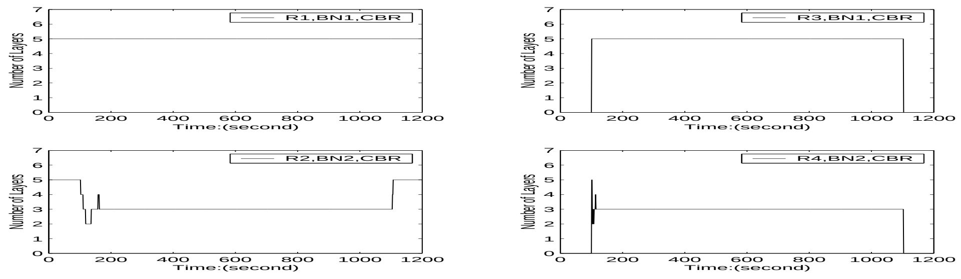


Fig. 2. Number of Layers: Two Competing CBR VM Sessions

The first thing we can see in these figures is that fairness is good at both bottlenecks, even if the two VM sessions start at different times. In the CBR case, R1 and R3 sharing the first bottleneck get 5 layers each, while R2 and R4 sharing the second bottleneck get 3 layers each. Although the first bottleneck can sustain 12 CBR layers (2.4M / 200k = 12), each VM source has only 5 layers, so R1 and R3 only get 5 layers each. At the second bottleneck, 1.2Mb/s is shared equally by two VM sessions, so R2 and R4 get 3 layers each. In the VBR case, the situation is a little complicated. With the VBR1 source, the bandwidth-sharing pattern is almost the same as that with CBR source, except that R2 and R4 only get 2 instead of 3 layers each. With the VBR2 source, both R1 and R3 get 4 layers, but R2 and R4 get different numbers of layers: 1 layer and 2 layers, respectively.

The reason that in the VBR2 case there is one layer difference between the two VM sessions at the second bottleneck is that the second bottleneck can only sustain 3 VBR2 layers (we will discuss why bottlenecks sustain a different number of layers with different VBR sources later). The 3-layer capacity can not be shared equally between the two VM sessions, so one VM session gets 2 layers while the other gets 1 layer. Generally, the granularity of the fairness in multi-layer multicast congestion control is one layer. When a bottleneck can only sustain a number of layers that can not be assigned equally among the sessions passing through it, one layer difference between some sessions is inevitable. We know end-to-end multi-layer video multicast congestion control schemes show serious unfairness between competing multicast sessions [9] [10]. Usually, later arrivers cannot grab a right share of bandwidth. In the RVMCC case, although VM session 1 and VM session 2 start at different times, they get a right share of the bandwidth at both bottlenecks.

The second thing we observed is that the number of layers at each bottleneck has good stability in all cases. The number of layers is almost constant in the CBR case, except the layer adjustment for the coming or the going VM session. In the VBR case, although there is some layer adjustment at each bottleneck, the layer adjustment only occurs in one VM session at each bottleneck. The other session still has a stable number of layers. Furthermore, in the session with layer adjustment the adjustment is rare and only goes in one direction from a basic level. At R1 in Fig3, the adjustment is from the basic level of 5 layers to 4 layers and then goes back to the basic level of 5, while at R4, the adjustment is from the basic level of 2 layers to 3 layers and then goes back. In RVMCC, higher variance of traffic only causes a less number of layers sustained by the bottleneck but not an unstable number of layers. This can be observed in Fig.4.

The third thing we can conclude from these figures is that higher variance of VM sources usually renders lower throughput at bottlenecks. With the VBR1 source, only 4 layers are sustained by the second bottleneck, but it sustains 6 CBR layers. With the VBR2 source, which has a higher variance than the VBR1 source, only 3 layers are sustained at the second bottleneck and the first bottleneck also only sustains 8 layers. This is due to the varying-rate characteristic of VBR sources. Without enough bandwidth reserved for a VBR source at a bottleneck, at its peak rate the VBR source may cause heavy losses and consequent layer-blocking at the bottleneck. The higher the variance, the more the bandwidth needed to be reserved for the peak rate. So higher variance usually causes lower throughput. A lower number of layers for VBR sources with higher rate variance is also necessary for maintaining fairness with TCP sessions, since frequent losses may throttle TCP traffic.

Lastly, the convergence is clear in all cases. With an incoming session or an outgoing session, other sessions respond appropriately to release or claim bandwidth. This can be observed in all figures.

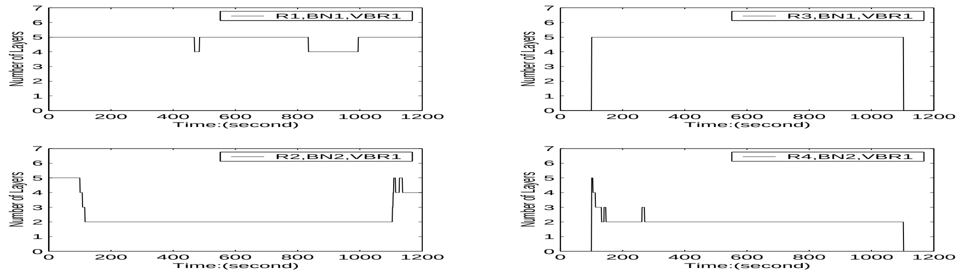


Fig. 3. Number of Layers: Two Competing VBR1 VM Sessions

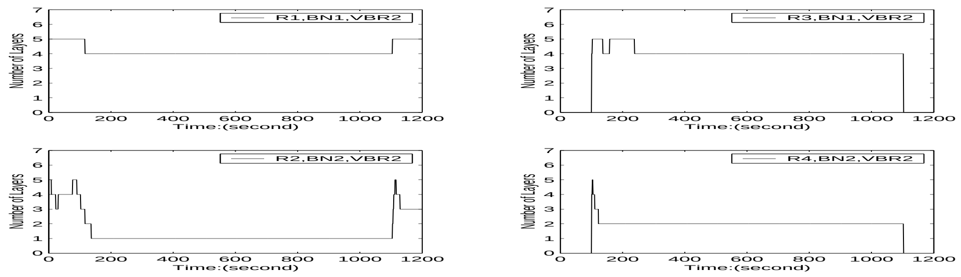


Fig. 4. Number of Layers: Two Competing VBR2 VM Sessions

Scenario Two: A TCP Session Joining Existing VM Sessions In this scenario, how VM sessions respond to a new TCP session is observed. A TCP session starts 100 seconds later after two VM sessions started. The simulation results are shown in Fig.5, Fig.6, Fig.7 and Fig.8. In these figures, we can see that the good stability and convergence characteristics observed in scenario 1 are still preserved in this scenario. We will focus on the fairness issue in the following text.

In the CBR case, in Fig.5 we can see that R1 and R3 get 5 layers each at the first bottleneck, but the link-equal-share at the first bottleneck is 800Kb/s (2.4Mb/s among 3 sessions), which means less than 5 layers for each session. This can be explained if we have a look at the second bottleneck. Still in Fig.5, we see that R2 and R4 get 2 layers each at the second bottleneck, which is just the link-equal-share there (6-layer capacity for 3 sessions). So the TCP session also gets a share of 2 layers (400Kb/s) at the second bottleneck. Because of the restriction at the second bottleneck, the TCP session can only consume at most 400Kb/s at the first bottleneck. This means that 2Mb/s out of 2.4Mb/s is left for the two VM sessions at the first bottleneck. If Max-Min fairness is

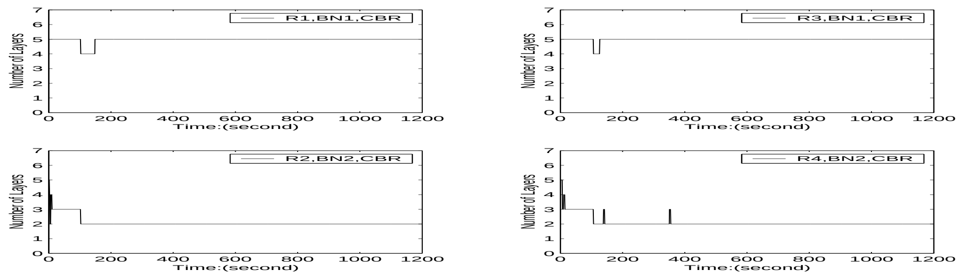


Fig. 5. Number of Layers: TCP Session Joining CBR VM Sessions

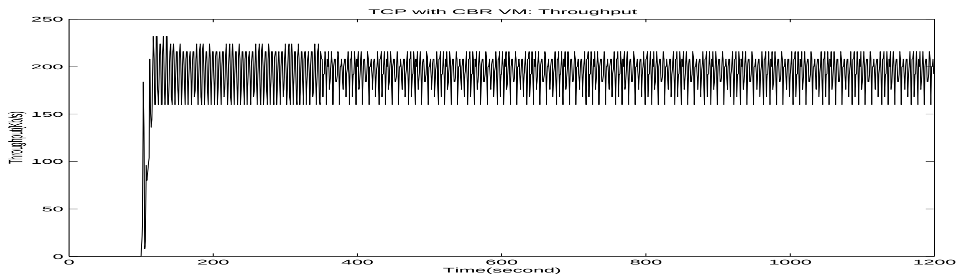


Fig. 6. TCP Throughput: TCP Session Joining CBR VM Sessions

approached, the two VM sessions should get 1Mb/s each at the first bottleneck. This is just what shown in Fig.5. But in Fig.6 we can find that the TCP session only achieves an average throughput much less than 400Kb/s. The reason is that TCP traffic is also a kind of VBR traffic.

In the VBR case, in Fig.7 we see that at the second bottleneck the two VM sessions have a one-layer difference. This is becuase the VM-total-link-equal-share there (400Kb/s x 2 = 800Kb/s) can only sustain 3 VBR1 layers due to the varying rate of the layers. In Fig.8, the TCP session achieves an average throughput almost the same as that in Fig.6. So there is still about 2Mb/s bandwidth left for the two VM sessions at the first bottleneck because of the restriction at the second bottleneck for the TCP session. Because the two VM sessions have VBR1 sources, each of them only achieves a throughput of 4 layers there.

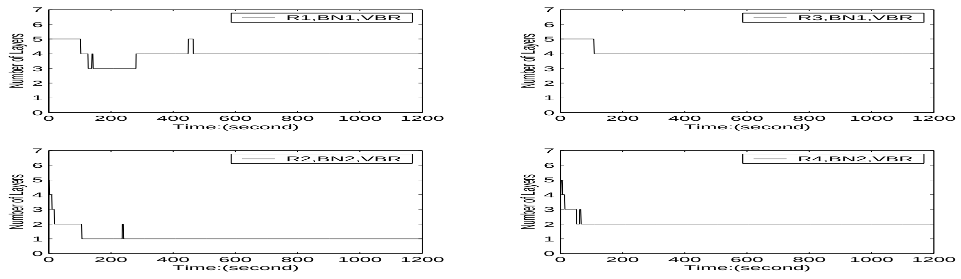


Fig. 7. Number of Layers: TCP Session Joining VBR1 VM Sessions

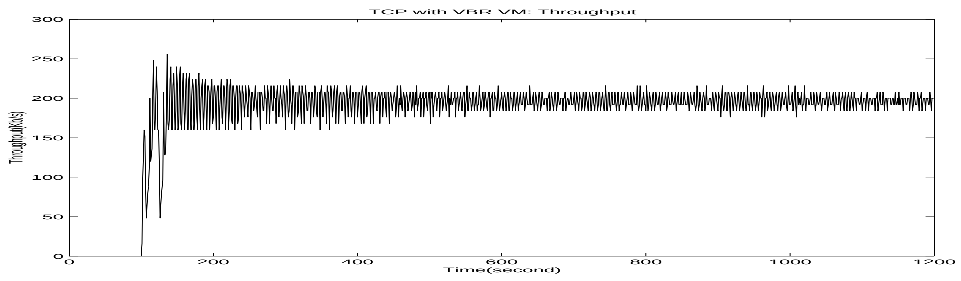


Fig. 8. TCP Throughput: TCP Session Joining VBR1 VM Sessions

In summary, in this scenario of TCP joining VM sessions, VM sessions still have stable numbers of layers and the convergence is still stable and clear. Furthermore, Max-Min fairness is closely approached at bottlenecks.

Scenario Three: VM Sessions Joining the Existing TCP Session In this scenario, VM sessions joining an existing TCP session is tested. VM session 1 and VM session 2 start 100 seconds later after the TCP session started. The simulation results are shown in Fig.9, Fig.10, Fig.11 and Fig.12.

In fact, this scenario is very similar to the last one, except that the VM sessions start later instead of earlier than the TCP session. If we compare figures from 9 through 12 with figures from 5 through 8, respectively, we can find that when all three sessions are alive the bandwidth-sharing patterns are almost the same in each pair of figures. For example, we can have a look at Fig.5 and Fig.9. In Fig.5, both VM sessions get 5 layers at the first bottleneck almost all the time. At the second bottleneck, they get 3 layers each before the TCP session arrives at the 100th second. After that they both get 2 layers each. In Fig.9, after the two VM sessions become alive at the 100th second, they get 5 layers each at the first bottleneck and get 2 layers each at the second bottleneck. So after the 100th second, which is the point that all 3 sessions become alive, the bandwidth-sharing pattern in Fig.5 is almost the same as that in Fig.9. The same thing happens for other pairs of figures.

In summary, in this scenario Max-Min fairness is still closely approached at both bottlenecks, although the two VM sessions become alive later than the TCP session. Both VM sessions still have good stability in number of layers. Furthermore, convergence is still steady and clear.

4 Summary

The RVMCC protocol attacks the multi-layer video multicast congestion control problem from a new direction: enriching the abstractions of the routing layer. It overcomes most of the inherent disadvantages of end-to-end multi-layer video multicast congestion control schemes, such as unfairness in sharing bandwidth and instability in throughput. It also avoids the complexity of most network-assisted congestion control schemes. No structure change of queuing, scheduling or forwarding of the current Internet is necessary for implementing this protocol. RVMCC is also scalable because there is no

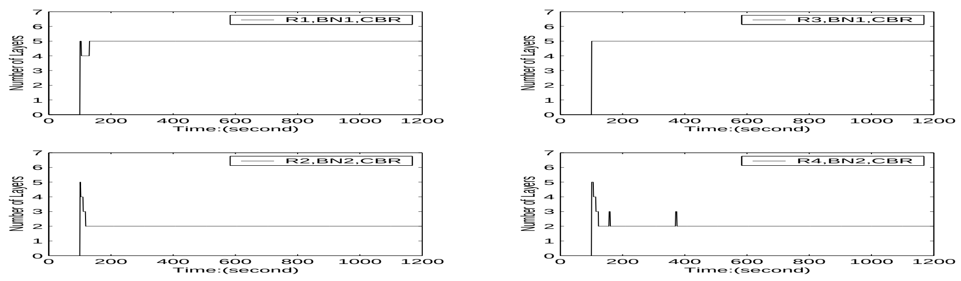


Fig. 9. Number of Layers: CBR VM Sessions Joining a TCP Session

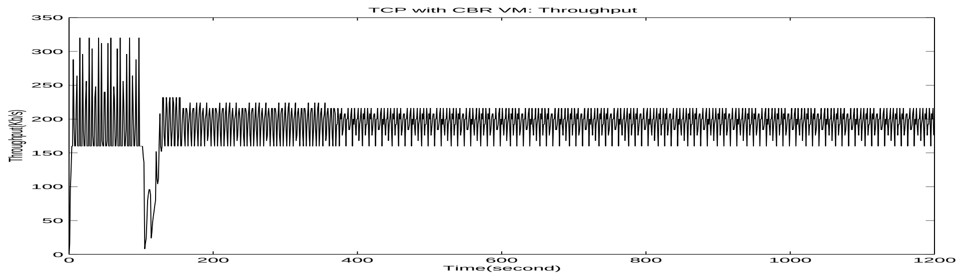


Fig. 10. TCP Throughput: CBR VM Sessions Joining a TCP Session

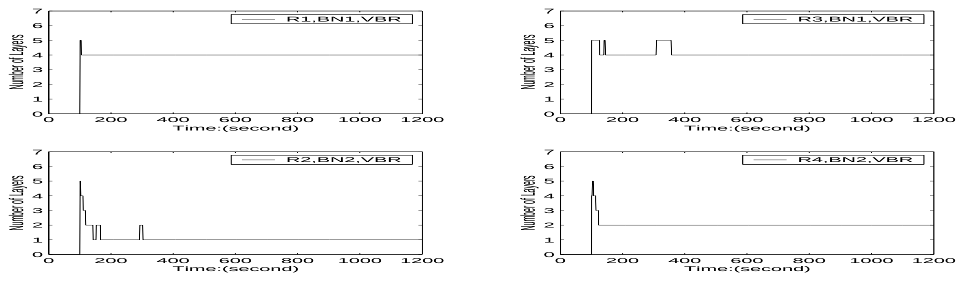


Fig. 11. Number of Layers: VBR1 VM Sessions Joining a TCP Session

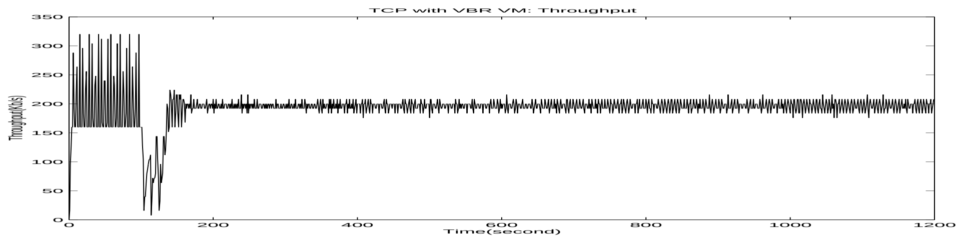


Fig. 12. TCP Throughput: VBR1 VM Sessions Joining a TCP Session

communication overhead for RVMCC protocol. Control actions in a router or receiver are independent of those in other routers and receivers. In end-to-end multicast congestion control schemes, communication overhead or necessary dependence of actions among receivers usually restricts their scalability. All of these characteristics give the RVMCC protocol the strength to be a good candidate for a standardized multi-layer video multicast congestion control protocol.

References

1. S. Deering ”Internet multicast routing: State of the art and open research issues,” Multimedia Integrated Conferencing for Europe (MICE) Seminar at the Swedish Institute of Computer Science, Stockholm, Oct. 1993.
2. N. Chaddha and A. Gupta ”A frame-work for live multicast of video streams over the Internet,” Proceedings of the IEEE International Conference on Image Processing , Lausanne, Switzerland, Sept. 1996.
3. L. Delgrossi, C. Halstrick, D. Hehmann, G. Her-rtwich, O. Krone, J. Sandvoss and C. Vogt ”Media scaling for audiovisual communication with the Heidelberg transport system,” Proceedings of ACM Multi-media ’ 93 Aug. 1993
4. D. Hoffman and M. Speer ”Hierarchical video distribution over Internet-style networks,” Proceedings of the IEEE International Conference on Image Processing Lausanne, Switzerland, Sept. 1996.
5. Mccanne, S. and Vetterli, M. ”Joint source/channel coding for multicast packet video,” Proceedings of the IEEE International Conference on Image Processing Washington, DC, Oct. 1995.
6. T. Turletti and J.-C. Bolot ”Issues with multicast video distribution in heterogeneous packet networks,” Proceedings of the Sixth International Workshop on Packet Video Portland, OR, Sept. 1994.
7. S. McCanne, V. Jacobson and M. Vetterli, ”Receiver-driven Layered Multicast,” Proceedings ACM SIGCOMM 96, August 1996.
8. L. Vicisano, L. Rizzo and J. Crowcrof, ”TCP-like Congestion Control for Layered Multicast Data Transfer,” Proceedings of IEEE INFOCOM, San Franciso, March 1998.
9. R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson and C. Sreenan. ”Stability and Fairness Issues in Layered Multicast,” Proceedings of the NOSSDAV ’99, June 1999.
10. A. Legout and E. W. Biersack, ”Pathological Behaviors for RLM and RLC,” Proc. of NOSSDAV’00, pp. 164–172, Chapel Hill, North Carolina, USA, June 2000.
11. J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter and W. Shaver. ”FLID-DL: Congestion Control for Layered Multicast,” Proceedings of NGC 2000, pages 71–81, November 2000.
12. S. Sarkar and L. Tassiulas, ”Back Pressure Based Multicast Scheduling for Fair Bandwidth Allocation;” Proceedings of IEEE Infocom April, 2001
13. N. Shacham, ”Multipoint communication by hierarchically encoded data,” Proc. IEEE INFOCOM’92, pp. 2107-2114, May 1992.
14. R. Gopalakrishnan, J. Griffoen, G. Hjalmtysson, C. Sreenan, and S. Wen, ”A Simple Loss Differentiation Approach to Layered Multicast,” Proc. IEEE INFOCOM’00, Mar 2000.
15. M. Yajnik, J. Kuros and D. Towslcy, ”Packet Loss Correlation in the MBone Multicast Network,” In Proceedings of IEEE Global Internet, November 1996.