Topology Sensitive Congestion Control for Real-Time Multicast

Srinivasan Jagannathan, Kevin C. Almeroth, and Anurag Acharya
Department of Computer Science
University of California
Santa Barbara, CA 93106-5110
{*jsrini, almeroth, acha*}*@cs.ucsb.edu*

Abstract

The multicast-based distribution of streaming media is conjectured to be an important component of future network services. Congestion control has emerged as a major hurdle in the large-scale deployment of such services. Heterogeneity in transmission media, end-systems, and traffic flows have significantly increased the complexity of the problem. Researchers have sought to address the problem through such mechanisms as layered encoding and TCP-style formulae. The end-to-end *paradigm has been the guiding principle for these approaches. In parallel, many tools are being developed that provide snapshots of network internals. Of interest to us are tools that accurately construct or estimate the topology of a multicast tree. In this paper, we explore whether these tools can assist in providing congestion control for real-time multicast. Through this exercise we find that the multicast tree topology, if available, can be used to regulate the flow of streams. In this paper, we seek to understand the additional benefits of using this information by focusing on the extreme case where the complete tree topology is known. We develop an algorithm which uses this information and layered streams to provide robust congestion control. We evaluate our algorithm using* ns*, the network simulator. Our results show that our algorithm is robust and converges to "fair" subscription levels for each receiver.*

1. Introduction

Streaming media is fast becoming the most popular media type being delivered in the Internet. The Internet, originally designed for data transport, is now increasingly being used to deliver multimedia services. IP Multicast efficiently supports this type of transmission by enabling a source to send a single stream to multiple recipients who explicitly want to receive the stream[1]. This yields many performance improvements and conserves bandwidth end-to-end.

However, the heterogeneity of the current Internet poses challenges to multicast transmission. The difference in available bandwidth makes it impossible to provide good service to all receivers using a single multicast stream. For instance, a receiver on the same 10Mbps Ethernet as the source can expect to receive a better quality of transmission than a receiver on a 33Kbps modem. Researchers have focused on satisfying these varying requirements by using multiple multicast streams. *Replicated streams* models as well as *layered streams* models have been proposed[2]. In the replicated streams model, the source sends multiple streams with the same content, but with different quality and bit rates. The layered streams model utilizes the ability of various video compression schemes to break up their output bit stream into multiple streams.

Researchers have frequently referred to the importance of knowing the tree topology in making optimal decisions[3, 4]. We corroborate this intuition by considering an illustrative example. Assuming a layered multicast model, receivers subscribe to a base layer and some enhancement layers. Assume that layer 1 requires a bandwidth of 32Kbps and every subsequent layer requires twice the bandwidth required by the previous layer. Using the topology in Figure 1, the receivers at nodes 3 and 4 can hope to receive layers 1 and $1, 2$ respectively. Suppose, the receiver at node 4 greedily tries to subscribe to one more layer. The receiver is bound to experience loss. In addition, the receiver at node 3 is also affected because of congestion at node 2. Protocols like Receiver-driven Layered Multicast(RLM) try to avoid such problems by multicasting receiver join experiments[5]. This however has the disadvantage that, receivers such as the one at node 5, who

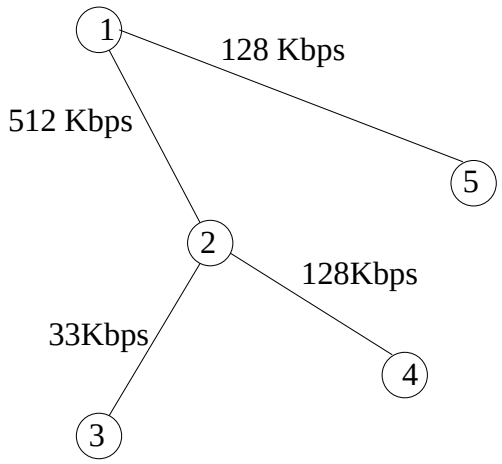

Figure 1. A simple multicast tree topology.

are topologically "unrelated" also get the multicast message. Knowledge of the tree topology would avoid such problems.

Our work tries to formalize the intuitive notion that "knowledge of tree topology is useful". We develop an algorithm that uses topology and loss information to suggest optimal subscription bandwidths. In this paper, we make an assumption that the entire tree structure is known to a "centralized controller agent". This agent then uses the algorithm we develop, and informs the receivers as to what layers they should subscribe. This architecture has obvious scalability limitations, but allows us to explore how tree topology can be used to adapt to congestion in the network. Our approach consists of acquiring the necessary information at periodic intervals (which we assume is feasible), running the algorithm and sending the results to receivers. We evaluate the operation of the algorithm by simulation using *ns*.

The remainder of the paper is organized as follows. We summarize related work in Section 2. Section 3 contains an assessment of the parameters of the problem and an enumeration of the principles for congestion control under these constraints. In Section 4 we develop a topology sensitive congestion control algorithm. In Section 5, we present a set of simulation results. The paper is concluded in Section 6.

2. Related Work

The problem domain we address has two parts: 1) efficient means of delivering multimedia content using multicast and 2) mechanisms to infer a multicast tree's topology. Li et al. present a comprehensive history of approaches to multicast-based multimedia content delivery[2]. Notable among these approaches is Receiver Layered Multicast (RLM) which uses a layered model of transmission[5]. In RLM, each receiver subscribes to a base layer and some enhancement layers. Join experiments are advertised and receivers undergo *shared learning* to find optimal subscription levels. Bajaj et al. propose a priority dropping scheme where the network, instead of receivers, decides optimal subscription levels[6]. Gopalakrishnan et al. propose a hybrid of receiver and network driven approaches wherein the receivers decide the priorities[7]. The key feature of all these approaches is that they abide by the *end-to-end* principle. If information internal to the network is used, then it is done "inside" the network, at routers. The receivers use only end-to-end information. While the end-to-end approach is appropriate for the unicast model, it has severe drawbacks when used for multicast. Coordination among multicast receivers—an important parameter for congestion control—is difficult to achieve if tree topology is not known. Ratnasamy and McCanne contend that tree inference using end-to-end measurements is sufficient for coordinating receivers[4]. However, inter-flow fairness cannot be addressed using their group formation protocol. Our approach is to develop an application-layer algorithm that relies on a topology-constructing tool to make suggestions about optimal subscription levels. An application level approach uses internal state of the network, but does not violate the end-to-end principle.

Critical to our work is the ability to discover the multicast tree topology. Recent research has focused on inference techniques to build a representation of the tree. Thaler et al. use cascaded SNMP router queries in their *mrtree* tool to provide a text based representation of the multicast tree topology[8]. Other work focuses on using the *mtrace* utility and various front-end interfaces. Levine et al. proposed the Tracer protocol, which uses the *mtrace* utility to deterministically organize multicast group receivers into a tree structure[9]. Makofske and Almeroth also use *mtrace* and RTCP packets to determine and visualize the multicast tree topology[3, 10]. Ratnasamy and McCanne infer logical multicast trees from shared loss characteristics[4].

3. Multicast Congestion Control

While TCP is sufficient for certain types of traffic, its congestion control algorithm is not appropriate for multimedia traffic. The TCP congestion control algorithm is de-

signed to reduce the transmission rate whenever congestion is detected[11]. TCP however cannot be used for real-time multimedia traffic because the characteristics of the traffic are different. While TCP is most suited to delay-tolerant and loss-sensitive traffic; multimedia traffic is loss-tolerant and delay-sensitive.

TCP congestion control involves interaction between the sender and the receiver and is feedback oriented. This is certainly not practical for multicast traffic because of the notorious "implosion" problem. If all the receivers send feedback to the sender, the root link in the tree will become congested. In addition, the sender becomes overwhelmed with the responses. UDP is therefore the preferred protocol to transmit multimedia traffic. UDP provides best effort delivery which is ideally suited for loss-tolerant traffic. In addition, there are no feedback mechanisms provided by UDP. UDP traffic has no cognizance of network congestion. Other protocols have therefore been developed on top of UDP to impart limited and scalable feedback and congestion awareness[2].

Research efforts have been directed to discover and quantify the TCP approach to congestion control, so that it can be applied to UDP traffic. Formulae have been developed to characterize TCP traffic and these have been used in UDP-based protocols[12]. These formulae rely on an estimate of round trip times from the sender, and on estimates of packet loss rates. These metrics run into an intuitive roadblock when it comes to defining round trip times for multicast. Because there are multiple receivers, attempts to define round trip times are nebulous at best. In addition, the characteristics of the streaming media traffic may not be conducive to TCP-like flow control. For instance, additive increase with multiplicative decrease, though practical for TCP, does not make sense for layered streaming media. Any attempts to simulate TCP-like behavior for such traffic requires a fundamental change in encoding techniques. In the absence of such a change, one cannot hope to simulate TCP like behavior and at the same time ensure "quality" in data reception. Our approach has been to take a liberal view towards TCP friendliness in the light of the following trends in Internet traffic[13]:

- Short-lived TCP connections form the bulk of Internet traffic
- Long-lived TCP connections are relatively few in number and declining
- The overall trend in the Internet traffic pattern (TCP versus UDP) has been fairly constant over time.

In contrast, we are focusing on multicast sessions whose duration is much longer. Coupled with the fact that multicast prunes may take time on the order of minutes[14], it is a reasonable possibility that individual TCP sessions will transmit their payload and cease to exist, by the time multicast congestion control kicks in.

Researchers have sought to model the TCP-like behavior in a multicast scenario by being receiver-oriented rather than sender-oriented. The receiver can obtain receiver-to-source characteristics like an estimate of round-trip time and available bandwidth. However, receiver-driven protocols suffer from a major setback: lack of co-ordination and cooperation. In general, it is not possible to achieve fairness without additional network machinery, even if all the receivers cooperate. McCanne et al. accept this as a problem, but mention that their approach will work well in consonance with network machinery that enforces this cooperation and fairness[5].

As an alternative to receiver-driven protocols, we examine the possibility of using a controller agent and the multicast tree topology to impose fairness on receivers. We hypothesize that the knowledge of the tree topology with some additional information such as packet loss rates at receivers can be used to determine a "fair" distribution of bandwidth utilization. While a single, centralized agent is likely to suffer the same scalability problems as a source-driven protocol, this problem can be solved hierarchically. Multiple agents, each concerned with different subtrees of the whole multicast tree, can be employed.

An important consideration is how easily the solution can be deployed. Much of the recent research has focussed on router-based solutions[6, 7, 15]. While these solutions may be elegant, they may not necessarily be easy to deploy. Our solution, which offers an application-layer agent using tools already in place, can be considered an attractive alternative. One caveat is that our solution does presume the existence of a topology constructing tool, which in practice may have its own scalability problems.

In general, we believe any topology-based congestion control algorithm should have certain properties. Using topology information we hope to develop an algorithm with the following characteristics:

1. Recognize occurrences of congestion

2. Allow receivers to explore available bandwidth

3. Adapt to transient traffic

4. Limit the maximum subscription of layers in a subtree to the minimum bottleneck bandwidth between any receiver in the subtree and the source.

4. The TopoSense Algorithm

To the extent in which we have examined other multicast congestion control algorithms and developed a set of desired properties, we now present our algorithm, called *TopoSense*. We discuss the algorithm in two parts. First, we outline the architecture of the algorithm. Second, we describe the algorithm and its operation.

4.1. TopoSense Architecture

TopoSense is assumed to be run at periodic intervals by controller agents. Multiple controller agents, each concerned with a different portion of the multicast tree, are located in different end-user networks. Each controller agent obtains regular updates of the tree topology in its domain and the layers traversing each part of that tree. This domain based solution has two primary advantages. First, it adds an element of scalability; congestion control can be managed on sub-trees instead of on the whole tree. Second, it is flexible to administrative heterogeneity. Thus a network using SNMP can use an SNMP-based topology discovery tool while some other network can choose not to use topology based congestion control at all. This architecture is illustrated in Figure 2.

The TopoSense instance used by each controller agent maintains an internal image of the current multicast topology and uses it for decision making. The TopoSense algorithm makes its decisions based on two important criteria. These are the multicast session topology and receiver packet loss rates. The algorithm does not require knowledge of link bandwidths, but knowledge of these would certainly lead to better performance. In general, a lack of knowledge about per-link capacity leads to the often-observed saw-tooth behavior in receiver bandwidth. This is a consequence of most algorithms needing to explore how much bandwidth is available and can be utilized. Receivers periodically attempt to increase the amount of received bandwidth. If congestion occurs receivers return to the previous rate. The algorithm also assumes that the average bandwidth of each layer is known beforehand. This assumption is reasonable as this can be advertised along with the multicast address of the layer.

4.2. TopoSense Algorithm

The TopoSense algorithm has three stages in addition to information retrieval from the network. In the first stage, congestion rules are used to label each tree node as CONGESTED or NOT−CONGESTED. This stage processes nodes in the tree breadth-first and bottom-up. There is also top-down pass in which uncongested children of congested parent nodes are also labeled as congested. The algorithm defines a node to be congested if:

- its parent is congested, or

- all its children have a packet loss rate greater than a threshold value $p_{threshold}$ and more than $\eta_{similar}$ percent of its children have packet loss rates which are close to the average packet loss rate of all the children.

Because loss information at internal nodes of the tree is difficult to obtain, the TopoSense algorithm makes some simple assumptions about packet loss at internal nodes. The packet loss rate at an internal node is taken to be the minimum packet loss rate of its children. The packet loss rate at internal nodes is computed in a bottom-up, breadth-first manner based on reported losses at the receiver. Receivers can report loss using mechanisms like (but not limited to) the Real-time Transport Control Protocol (RTCP)[16]. The intuition behind this mechanism is that if all the children of a node are congested, then all the children will have to reduce their bandwidth demands. This in effect means that the parent node reduces its demand to a level equal to the maximum demand of all its children (in case of the layered model used in this work), even if it is not congested.

In the second stage of the algorithm subscription demands are computed for each node in the tree. This stage proceeds in a bottom-up fashion. This allows the nodes representing the receivers to subscribe to some layers and the nodes representing upstream routers to aggregate all the demands of the downstream nodes. This stage uses each node's subscription history (the highest layer subscribed in the previous two intervals) and the congestion state history (the congestion state computed in the previous executions of TopoSense) to make decisions regarding future subscriptions.

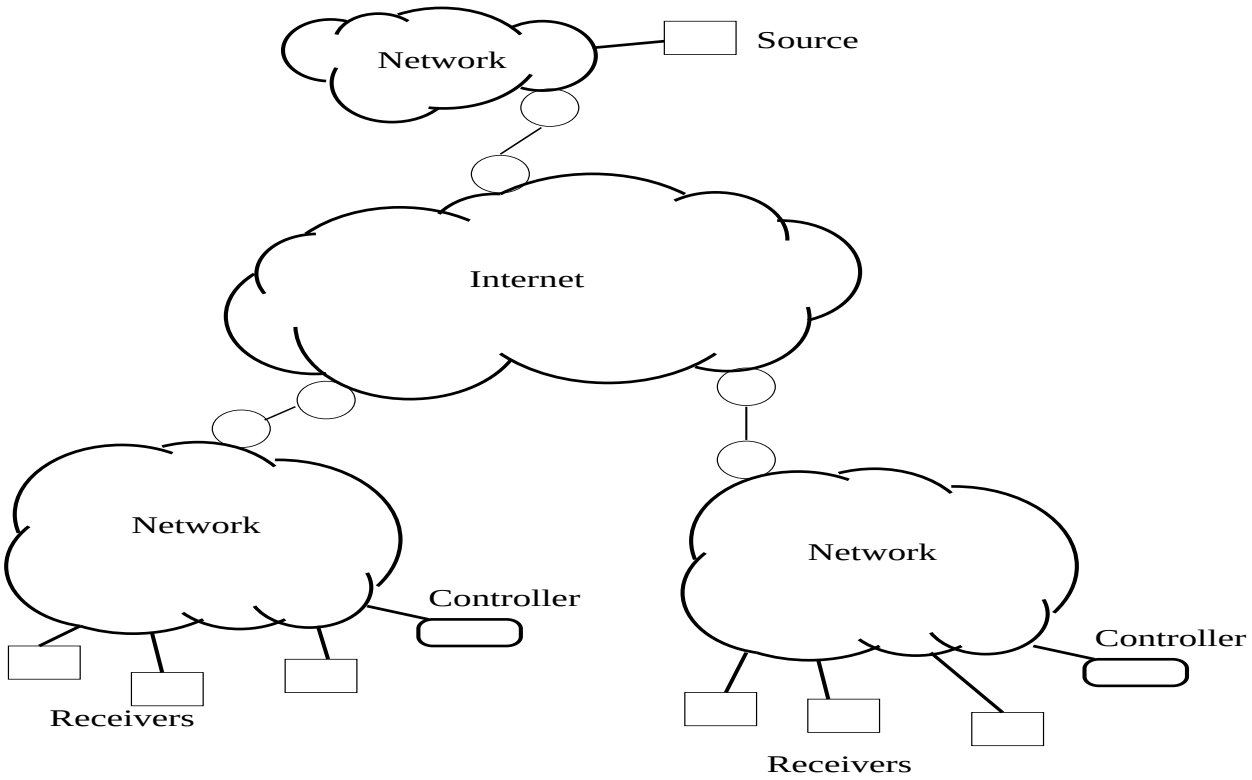

Figure 2. Scalability using a hierarchical control model.

The third stage is a top-down pass in which allocations are made for each demand. Once the allocation is made, the controller agent sends each receiver a message suggesting which layers it must subscribe.

The most critical part of TopoSense is the second stage where demands are computed. The decision making process can be represented as a table lookup indexed by the congestion state history and total advertised bandwidth subscribed in the past two intervals. The decision table is presented in Table 1. The congestion state history is presented as a 3 bit integer in the table where CONGESTED=1 and NOT−CONGESTED=0. The congestion states at times T_0, T_1 and T_2 are represented at bit positions 2, 1, and 0 respectively. The column, "BW Equality", represents the relationship of the total bandwidth received in interval $T_0 - T_1$ with respect to total bandwidth received in interval $T_1 - T_2$.

As an example, assume that TopoSense is run in intervals and the i^{th} interval is represented by T_i. TopoSense computes the demand for each node in the interval $T_2 - T_3$, using the congestion states computed at times T_0, T_1 and T_2. It also uses the total bandwidth of layers received in the intervals $T_0 - T_1$ and $T_1 - T_2$. If a node is labeled as CONGESTED at time T_2 and its parent is also CONGESTED, TopoSense assumes that the possible cause of losses at the child node is the congestion at the parent node. TopoSense then sets the demand of the child node to be the layers it was receiving in the interval $T_1 - T_2$. If the child node is labeled as CONGESTED but the parent node is not, TopoSense takes that into consideration when setting the level for the child node. In general, if the current node is an internal node and it has been labeled as CONGESTED, then the demand for layers is reduced such that the overall bandwidth demanded is less than the bandwidth demanded in the interval $T_1 - T_2$. If the node has been labeled as CONGESTED for successive intervals, the demand is reduced such that the total bandwidth demand is halved.

TopoSense increases the bandwidth demanded for leaf nodes (receivers) only when the congestion state history is NOT−CONGESTED successively for two intervals. However, if an increase in demand is followed by a CONGESTED state in the next interval, then TopoSense sets a backoff timer for the receiver. The demand for that receiver will not increase until the timer expires.

Leaf/Internal Node	Congestion State History	BW Equality	Action
Leaf	0	Lesser	Add next layer, if not backing off.
Leaf	1	Lesser	If loss rate is high, drop layer, set backoff timer
Leaf	2,4,5,6	Lesser	Maintain Demand
Leaf	3	Lesser	Reduce demand to supply in $T_0 - T_n$
Leaf	7	Lesser	Reduce Demand to half the supply in $T_0 - T_n$ Set the backoff timer
Leaf	0,4	Equal	Add next layer, if not backing off.
Leaf	1,2,5,6	Equal	Maintain Demand
Leaf	3,7	Equal	Reduce Demand to half the supply in $T_0 - T_n$ Set the backoff timer.
Leaf	0	Greater	Add next layer, if not backing off.
Leaf	1,2,4,5,6	Greater	Maintain Demand
Leaf	3,7	Greater	If loss is very high, then reduce demand to half the supply in $T_0 - T_n$
Internal Node	0,4	All Cases	Accept all demands of the child nodes
Internal Node	1,5,7	Greater	Reduce Demand to half the supply in $T_n - T_{2n}$
Internal Node	1,5,7	Equal, Lesser	Reduce Demand to half the supply in $T_0 - T_n$
Internal Node	2,3,6	All Cases	Maintain Demand

Table 1. Decision table for computing demand at each node at time T_2.

5. Evaluation of the TopoSense Algorithm

In this section, we outline the issues relevant to the performance of TopoSense. We also present some simulation results to illustrate robustness of the algorithm. While the simulations do not *prove* that TopoSense is robust, they do indicate that TopoSense behavior is consistent with our intuitions. We evaluate TopoSense using the following criteria:

1. **Robustness and convergence time**. Each receiver must quickly receive its maximum (fair) bandwidth.
2. **Adaptation and Recovery**. The receivers must adapt to noisy cross-traffic and recover quickly after the cross-traffic subsides.
3. **Inter-session fairness**. When there are multiple sessions traversing the same set of links, bandwidth should be fairly and fully utilized.
4. **Session Scalability**. The algorithm must scale to large numbers of receivers.
5. **Stability**. There must not be frequent changes in bandwidth received. Ideally, layers should only be removed when congestion occurs and layers should only be added when capacity exists.

We implemented the TopoSense algorithm described above in the network simulator *ns*. Our work uses a hierarchical source model[5]. Sources transmit a layered video session consisting of 6 layers. The base layer sends at a rate of 32Kbps, with the rate doubling for each subsequent layer. We consider constant bit rate as well as variable bit rate sources. We used the same method to generate variable bit-rate (VBR) traffic as proposed by Gopalakrishnan et al.[17]. For the base layer, n packets are transmitted in 1 second intervals. n has the value 1 with probability $1 - \frac{1}{P}$ and the value $PA + 1 - P$ with probability P. A is the average number of packets per interval and P is the peak-to-mean ratio. Peak-to-mean ratios in the range of 2 to 10 have been observed for VBR traffic[17]. The packet size is chosen to be 1000 bytes.

In this paper we present results only addressing the issue of robustness. The other issues mentioned above are beyond the scope of this paper. To evaluate robustness, we used the topologies illustrated in Figure 3. Topology A is a simple topology consisting of a single receiver and a source. This topology is used to evaluate if TopoSense converges to optimal subscription in the absence of any heterogeneity or competing traffic. Topology B is an extension of Topology A, and is used to evaluate scalability (in a limited sense).

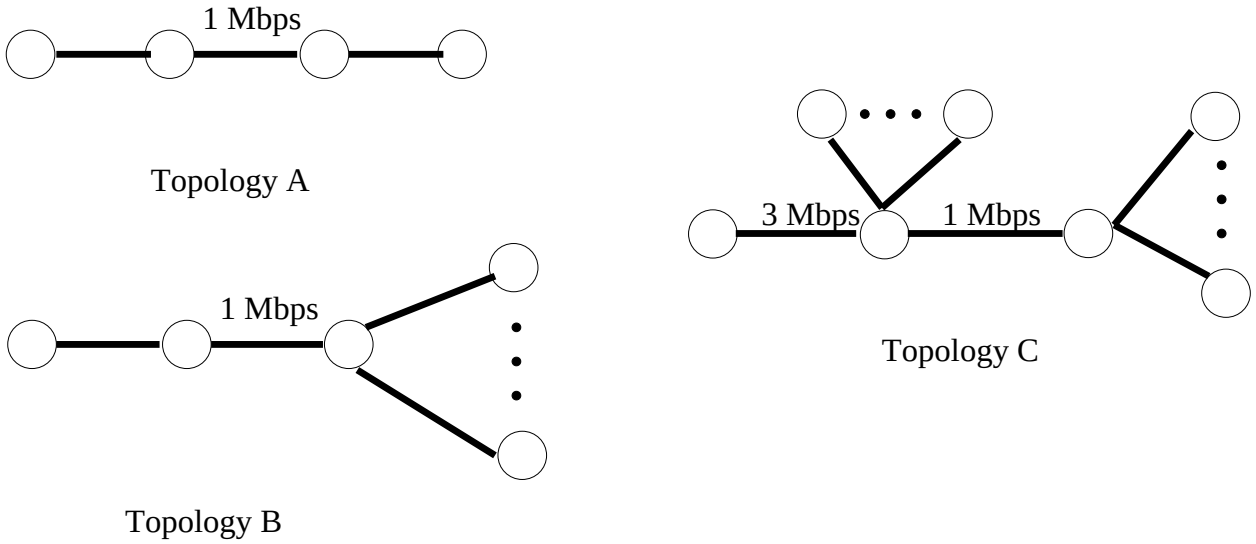

Figure 3. Simulation topologies.

The number of receivers attached to a node is increased and the subscription levels of all the receivers are monitored. Topology C is used to evaluate TopoSense behavior in a heterogeneous environment. There are two sets of receivers, each having different bandwidth constraints. The number of receivers in each set is increased and their subscription level monitored. To measure the performance of TopoSense, we define the following metric. Let $x_i(\Delta t)$ be the subscription level of receiver i in the interval Δt and y_i be the optimal subscription. Let $\parallel \Delta t \parallel$ be the length of the interval Δt. We define the relative deviation over the entire interval $\sum_{\Delta t}$ to be:

$$\frac{\sum_{\Delta t} \mid (x_i(\Delta t) - y_i) \times \parallel \Delta t \parallel) \mid}{\sum_{\Delta t} y_i \times \parallel \Delta t \parallel}$$

Intuitively, the smaller the relative deviation, the better the performance. It can be noted that once a stable state has been achieved, this metric can be made arbitrarily good by increasing the time interval. However, this is contingent on the system achieving a stable state. This metric is sufficient for our purposes because our objective is to illustrate that TopoSense is robust.

For the simulation parameters listed above, we ran numerous *ns* simulations. A sample result from each, for each topology, is shown in Figures 4, 5, and 6. The results in Figures 4, 5 and 6 show straightforward protocol behavior where each receiver joins the appropriate set of levels. The receivers in Topologies A and B have a bottleneck bandwidth of 1 Mbps and can receive only 5 layers (992 Kbps).

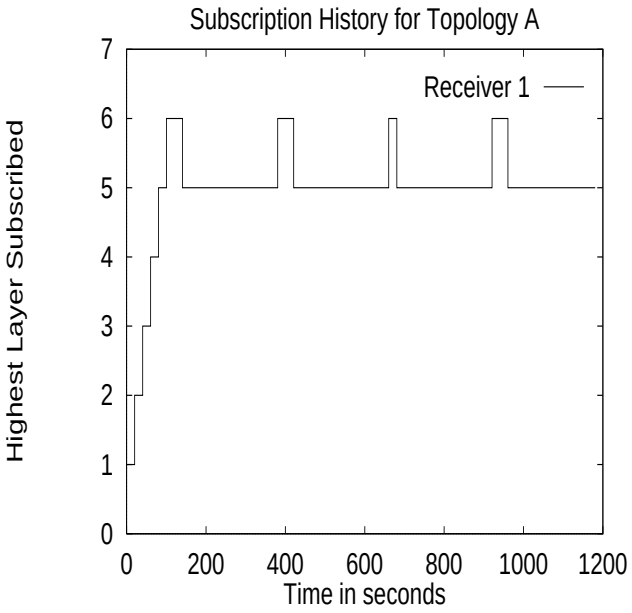

Figure 4. Topology A and VBR, P=3.

Some receivers in Topology C have a bottleneck bandwidth of 3Mbps and can receive all 6 layers, while other receivers have a bottleneck bandwidth of 1Mbps and can receive only 5 layers. This is illustrated in the subscription behavior shown in these Figures. The loss rate of the receivers for the simulations in Figures 5 and 6 are shown in Figures 7 and 8 respectively. The loss rates have been calculated using 10 second sliding windows. Receivers sharing a common bottleneck link show very similar loss rates. Join experiments by one of the receivers equally affects all other "related" receivers. When one of the receivers subscribes to a layer higher than the optimal level, the loss rates increase drastically. TopoSense is able to recognize and control the congestion only after some delay. This may appear counter-intuitive. However, in practice multicast prunes may take longer to take effect than the delay incurred because of TopoSense.

The average relative deviation from optimal subscription levels, over a period of 1200 seconds, for increasing number of receivers in Topologies B and C is shown in Figures 9 and 10 respectively. The average relative deviation for the constant bit-rate (CBR), VBR($P = 3$) and VBR($P = 6$) traffic is shown. One would expect this deviation to be very small and remain nearly invariant with increasing number of receivers. TopoSense behaves as expected. Topology B can have a maximum relative deviation of 0.8, while Topology C can have a maximum deviation in the range 0.8 to 0.83. These values correspond to the situation where the receivers subscribe to only one layer whereas the optimal is 5 layers or 6 layers. As can be seen, the results are invariant of heterogeneity constraints indicating that TopoSense suggests nearly optimal subscription levels to receivers in the above topologies. The behavior however is sensitive to the synchronization in join experiments. For instance, when we simulated a session in Topology B with 20 receivers, with random backoff intervals on join experiment failure, the loss rates were persistently high. This was because at least one of the receivers was performing a join experiment to the highest layer most of the time. All receivers incurred losses owing to these join experiments. On using a synchronized join experiment schedule (using back-off for internal nodes), this problem was eliminated. The burstiness of the traffic also appears to play a role in the behavior of TopoSense, when the number of receivers is high. This is seen in the higher relative deviation for VBR($P = 6$) traffic in Figure 9.

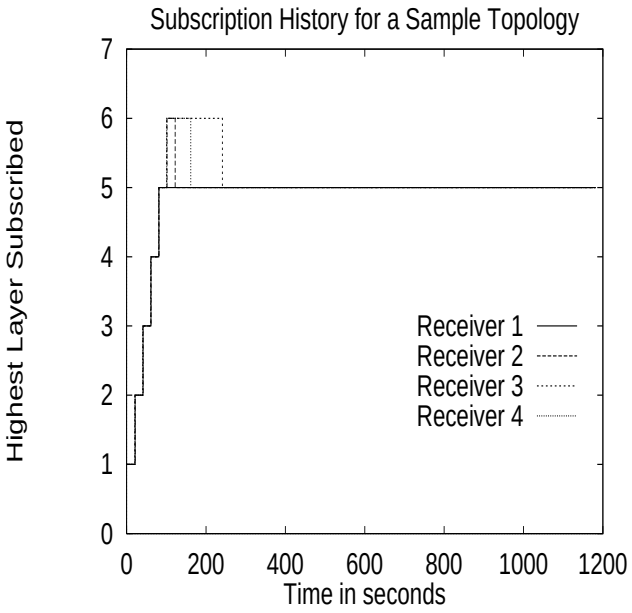

Figure 5. Layer Subscription: Topology B with 4 receivers and VBR, P=3.

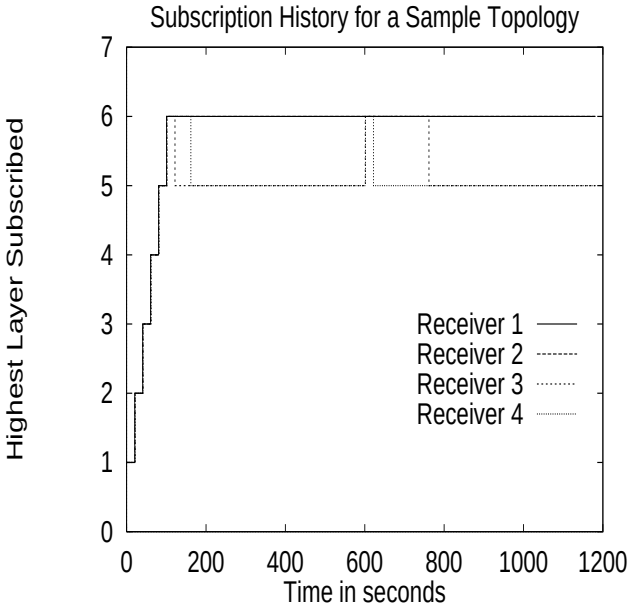

Figure 6. Layer Subscription: Topology C with 4 receivers and VBR, P=3.

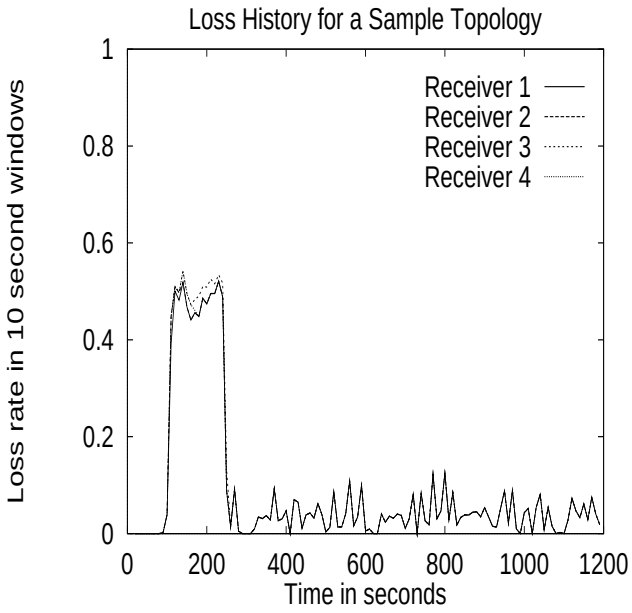

Figure 7. Loss History: Topology B with 4 receivers and VBR, P=3.

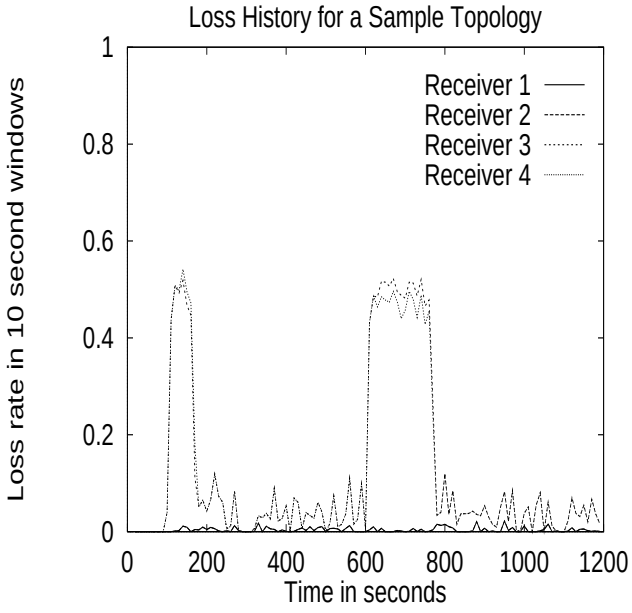

Figure 8. Loss History: Topology C with 4 receivers and VBR, P=3.

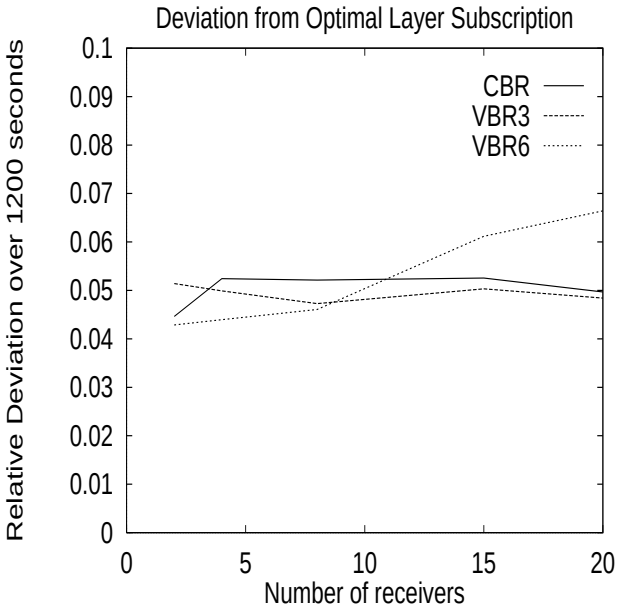

Figure 9. Relative Deviation: Topology B with different types of traffic

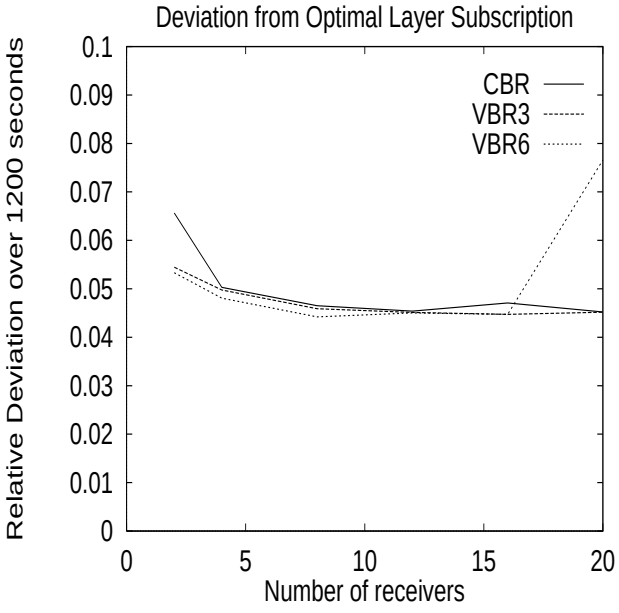

Figure 10. Relative Deviation: Topology C with different types of traffic

6. Conclusions

In this paper, we have explored using multicast tree topology information to assist in controlling real-time multicast traffic flow. We assessed the the key features of multicast traffic and identified the characteristics of a topology sensitive congestion control algorithm. Comparing the TCP model and existing receiver-based congestion control algorithms we offer insight into why these protocols do not sufficiently solve the problem. Using these insights we have developed an algorithm to suggest optimal subscription levels for receivers. We implemented the algorithm in *ns* and ran simulations to evaluate its robustness. The algorithm appears to be robust and converges to nearly optimal solutions in the basic tests run to-date. While our work identifies topology-awareness as a useful tool to address multicast congestion, significantly more work needs to be done to analyze and address the challenges in ascertaining topology in a timely manner and using it in a scalable fashion.

References

[1] K. Almeroth, "The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment," *IEEE Network*, January/February 2000.

[2] X. Li, M. Ammar, and S. Paul, "Video multicast over the Internet," *IEEE Network*, April 1999.

[3] D. Makofske and K. Almeroth, "MHealth: A real-time graphical multicast monitoring tool for the MBone," in *Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, (Basking Ridge, New Jersey, USA), June 1999.

[4] S. Ratnasamy and S. McCanne, "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements," in *IEEE Infocom*, (New York, New York, USA), March 1999.

[5] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *ACM Sigcomm*, (Stanford, California, USA), pp. 117–130, August 1996.

[6] S. Bajaj, L. Breslau, and S. Shenker, "Uniform versus priority dropping for layered video," in *ACM Sigcomm*, (Vancouver, British Columbia, Canada), pp. 131–143, September 1998.

[7] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, C. Sreenan, and S. Wen, "A simple loss differentiation approach to layered multicast," in *IEEE Infocom*, (Tel Aviv, ISRAEL), March 2000.

[8] D. Thaler and A. Adams, *Mrtree*. Merit Network, Inc. and University of Michigan. Available from http://www.merit.edu/net-research/mbone/mrtree_man.html.

[9] B. Levine, S. Paul, and J. Garcia-Luna, "Organizing multicast receivers deterministically by packet-loss correlation," in *ACM Multimedia*, (Bristol, ENGLAND), September 1998.

[10] D. Makofske and K. Almeroth, "Real-time multicast tree visualization and monitoring," *Software–Practice & Experience*, 2000.

[11] V. Jacobson, "Congestion avoidance and control," in *ACM Sigcomm*, (Stanford, California, USA), pp. 314–329, August 1988.

[12] M. Mathis, J. Semke, J. Mahdavi, , and T. Ott, "The macroscopic behaviour of the TCP congestion avoidance algorithm," *ACM Computer Communication Review*, vol. 27, July 1997.

[13] S. McCreary and K. Claffy, "Trends in wide area ip traffic patterns, a view from ames internet exchange." http://www.caida.org/outreach/papers/AIX0005.

[14] B. Williamson, *Developing IP Multicast Networks, Volume I (Fundamentals)*. Indianapolis, Indiana, USA: Cisco Press, 1999.

[15] A. Legout and W. Biersack, "PLM: Fast convergence for cumulative layered multicast transmission schemes," in *ACM Sigmetrics*, (Santa Clara, California, USA), June 2000.

[16] H. Schulzrinne, S. Casner, R. Frederick, and J. V., "RTP: A transport protocol for real-time applications." Internet Engineering Task Force (IETF), RFC 1889, January 1996.

[17] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, and C. Sreenan, "Stability and fairness issues in layered multicast," in *Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, (Basking Ridge, New Jersey, USA), June 1999.