TECHNICAL RESEARCH REPORT

On Satellite Multicast to Heterogeneous Receivers

by Apinun Tunpan, M. Scott Corson

CSHCN TR 2001-16
(ISR TR 2001-25)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

On Satellite Multicast to Heterogeneous Receivers

Apinun Tunpan and M. Scott Corson
Institute for Systems Research, University of Maryland, College Park, MD 20742 USA*†
{ apinun,corson }@isr.umd.edu

February 14, 2001

Abstract

We propose a framework for single-source, satellite-based multicast dissemination of bulk files. The framework trades off between reception delay and bandwidth usage and coexists with terrestrial background network traffic; specifically TCP traffic utilizing a short-term congestion control mechanism. The framework consists of two major components: 1) a multicast rate scheduling mechanism that uses long-term, end-to-end multicast packet survival statistics in order to deal with the bandwidth-delay trade-off issue, and 2) a fair queueing algorithm that regulates the points where multicast traffic from the satellite meets terrestrial background traffic. We show through simulation the performance of this framework under a number of scenarios.

1 INTRODUCTION

Multicast is an efficient way to transmit the same data from a single source to multiple receivers. Two of the current research issues in multicast involve with how to deal with receiver heterogeneity and how to achieve fair bandwidth sharing between multicast traffic and other existing network traffic. Regarding receiver heterogeneity, some research focuses on multiple channel or *multi-layered* approaches (e.g. [VRC98, RJL+00, ST00]). In a multi-layer approach, each receiver subscribes to a subset of channels (or layers) in such a way that the total capacity of the subscribed channels does not exceed the receiver's capacity. Such multi-layered approaches can be applied to the transmission of both continuous data streams (e.g. video) in which receivers can tolerate losses, and reliable multicast (e.g. bulk files) in which complete reception of the whole file is a requirement. When applying the multiple-channel approach to the case of finite-sized bulk file, we generally see larger transmission redundancy (e.g. in [BKTN98]) in a trade-off to achieve smaller reception delays, or are faced with a channel synchronization problem unless the data is wisely partitioned and transmitted (e.g. [DAZ99]).

*Prepared through collaborative participation in the Advanced Telecommunications & Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0002.

†The views and conclusions in this document are those of the authors and shoul d not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government.

We recently proposed a single-source, single-channel (i.e. singly-layered) long-term, rate scheduling approach for multicasting bulk files over a hybrid satellite-terrestrial network [TC00]. The 'long-term' rate scheduling approach is designed for an environment in which the propagation delay is significant, and the capacity of the feedback channel is limited. A receiver cannot send a negative acknowledgement (NACK) for every lost packet, but rather reports to the multicast source the long-term packet survival statistics conditioned on each of the (discrete) source's transmission rates. Utilizing these packet survival statistics and a forward error correction (FEC) capability, our approach effectively deals with receiver heterogeneity by adjusting the 'length' of each transmission rate in the source's transmission rate sequence. One of the unsolved problems, however, is that when the satellite multicast traffic (from the sky) merges with existing local terrestrial traffic, the two might not share the available terrestrial bandwidth fairly, especially when the terrestrial traffic uses TCP's congestion control mechanism whose effective throughput can drop sharply in a very short period in response to network congestion that the long-term multicast rate scheduling mechanism might have caused.

We assume that each satellite earth station receiving multicast data from the satellite is located very near (i.e. a few hops away) to the multicast receivers. This configuration is reasonable for many deployment scenarios as the satellite multicast traffic avoids the congested network routes in the terrestrial network. We also assume that there are some background TCP traffic streams originating from some nearby terrestrial sources, also being received by multicast receivers, and thus sharing a portion of the terrestrial network links with the multicast traffic. Recent work such as [GB00] has called for network support for multicast in heterogeneous environments. In a similar fashion this paper focuses on a simple use of fair queueing algorithms at the routers where the multicast traffic meets terrestrial traffic near to the receivers (i.e. at the very edge of the networks), as shown in Fig 1. The fair queueing algorithm blocks the excess part of the multicast traffic, protects the TCP traffic from starvation, and at the same time makes the multicast network appear heterogeneous (from the sender's perspective) due to both the presence of the terrestrial traffic and the varying link capacity. The multicast rate scheduling mechanism we utilize then handles this heterogeneity.

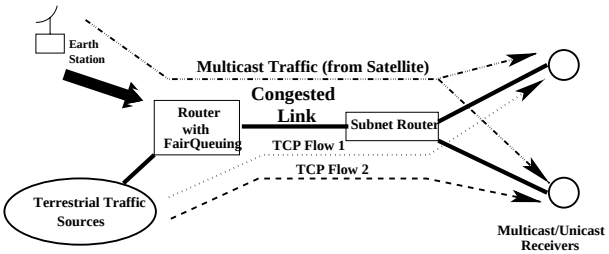


Figure 1: Fair Queueing in Satellite Multicast

2 THE MULTICAST RATE SCHEDULING PROBLEM

In this section, we describe the long-term multicast rate scheduling problem we study. We assume that a single multicast source can transmit at some rate r selected from a finite discrete set $\mathcal{R}$ (e.g. $\mathcal{R} = \{512, 256, 128, 64\}Kbps$). The source has a single, finite size bulk file ready to be transmitted at a time. If there are two or more files ready to transmit, they can be logically combined into one bulk file. Let the multicast source use Reed-Solomon erasure (RSE) coding (e.g. [Mac97]) and let the bulk file be encoded into $W \geq 1$ RSE-based FEC blocks. Each FEC block consists of h data packets and c parity packets – all of the same length. We assume a multicast tree that has a root at the sender and has a finite number M of

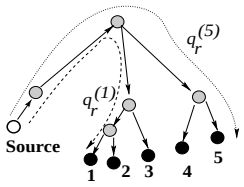


Figure 2: A Sample Multicast Tree Model (M=5).

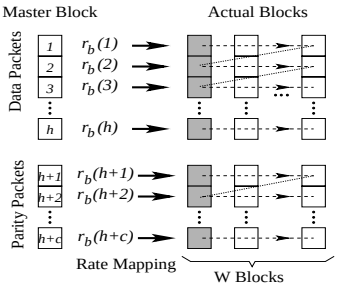


Figure 3: Mapping the master block to the actual data blocks

receivers (e.g. Fig. 2). We assume that, when perceived from an individual receiver (i.e. a one dimensional view), each packet transmitted by the multicast source survives the network loss and arrives at the receiver independently. Each receiver $m \in \mathcal{M}, \mathcal{M} = \{1, 2, \ldots, M\}$, monitors the long-term, end-to-end multicast packet survival probability $q_r^{(m)}$ at each of the source transmission rate $r \in \mathcal{R}$. We assume that $q_r^{(m)} \in (0, 1]$ is relatively time-invariant for our scheduling period – this is generally true if there is no route change and any other average network background traffic remains the same. The multicast source initially transmits a number of probe packets at each rate $r \in \mathcal{R}$ so that the survival estimator of $q_r^{(m)}$ can be initialized. Each receiver periodically reports its packet survival statistics to the multicast source to aid the rate schedule computation. In computing a rate schedule, each receiver has weight one. For scalability, a subset of receivers that have similar packet survival characteristics (e.g. inside a subnet) may be represented by a single representative who reports the packet survival statistics on the subset's behalf, having weight equals to the number of receivers it represents.

The multicast source transmits packets from each of the W FEC blocks interleavingly and progressively as appearing on the right of Fig. 3 (each FEC block is shown vertically). This technique is well-known and is used to suppress the effect from burst losses, if any, that might have violated our independent packet survival assumption. To successfully decode an RSE-based FEC block, a receiver must receive at least h distinct packets, either data or parity, from the same FEC block. To allow a systematic way to compute a rate schedule, we introduce the notion of the master block, as shown on the left of Fig. 3. The master block also has h data and c parity packets. Relying on the receivers' packet survival statistics, the independent packet survival assumption and the time-invariant assumption, the rate scheduling mechanism can then determine the appropriate amount of the master block's parity packets c and the master block's rate assignments $r_b(j)$; $j = 1, 2, \ldots, h, h+1, \ldots, h+c$ subject to a pre-defined reception reliability requirement and minimizing a composite cost function which has two competing cost components: the normalized bandwidth cost component and the normalized aggregate delivery delay cost component. The composite cost function also has a 'knob' $\phi \in [0, 1]$ parameter which controls the relative importance between the two competing cost components. Each of the master block's rate assignments $r_b(j)$, $j = 1, 2, \ldots, h + c$, can then be actuated on the j-th 'row' of each of the W actual FEC block as shown in Fig. 3.

In [TC00], we showed, per receiver, the probabilistic relationships between the master block's reception reliability and the bulk file's reception reliability. When a data block is sufficiently large (i.e. $h \gg 1$), we also observed that the independent packet survival assumption allows us to approximate the (virtual) master block's reception status at each receiver by the Gaussian distribution because a special case[1] of the central limit theorem can be applied. Based on this Gaussian approximation technique, the approximate solutions

[1] the 'uniformly bounded' case

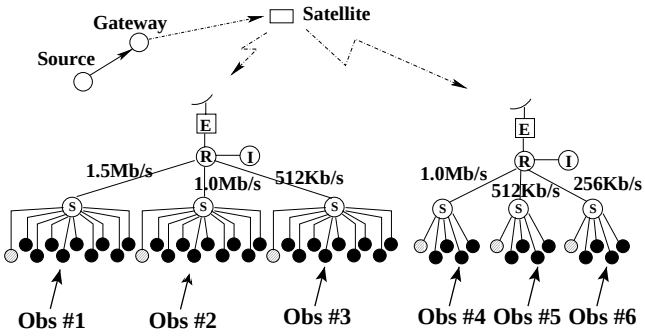


Figure 4: Topology Used in The Experiments

to two special cases of the rate scheduling problem were also proposed. In the first case, the rate scheduling algorithm simply selects one rate $r \in \mathcal{R}$ which best suits the composite cost function. In the second case, the rate schedule utilizes each and every rate $r \in \mathcal{R}$ in a decreasing order: the rate scheduling algorithm determines the number of master block packets to be transmitted at each rate $r \in \mathcal{R}$. We have found that the second algorithm, although requiring more computational overhead, is more preferable than the first algorithm in providing bandwidth-delay trade off for heterogeneous receivers. We use the second algorithm for the rest of this paper.

3 THE FAIR QUEUEING ALGORITHMS

Two broad types of fair queueing algorithms have been proposed. The first type of fair queueing algorithms requires some per-flow state to be maintained at the routers (e.g. "Deficit Round Robin" or DRR [SV95]). The second type of fair queueing algorithms improve scalability by dividing routers into *edge* routers and *core* routers. The edge routers, which reside near the edge of the network, maintain per-flow states, and tag each packet by certain value that represents the flow properties (e.g. rate) before forwarding the packet into the network core – where the core routers compare only the information on the packet's tag to decide whether to drop or forward the packet (e.g. "Core-Stateless Fair Queueing" or CSFQ [SSZ98] and $\mathcal{T}$UF[CD99]).

Recall from Section 1 that the satellite multicast traffic merges with the terrestrial network traffic near the edges of the network. Using the fair queueing algorithms which maintain per-flow state is still reasonable because fewer active flows exist at the network edges than those inside the network core. We focuses on two fair queueing algorithms: the well-known DRR [SV95] and its extension DRR+ [HMMM00]. DRR has a theoretical fairness for flows with different packet lengths. DRR+ extends DRR by replacing the DRR's per-flow FIFO mechanism by a per-flow random early detection (RED) mechanism. DRR+ was shown in [HMMM00] to provide better fairness among reno TCP traffic flows than its DRR counterpart. In this paper, however, we use DRR and DRR+ mainly both to block excess part of the long-term multicast traffic from reducing background Reno TCP throughput and to create a favorable operating environment for our long-term rate scheduling mechanism.

4 EXPERIMENTS AND RESULTS

We use the network simulator (*ns-2*) to study the performance. The topology used in our simulation is depicted in Fig. 4; (E) denotes an earth station, (R) denotes a router, (S) denotes a sub-net router, and (I) denotes the source of the terrestrial background traffic. Each receiver, denoted by either black or shaded leaf nodes, receives both multicast traffic and unicast FTP traffic; the unicast FTP traffic transmits infinite-length data, utilizes reno TCP and originates from the corresponding (I) node. A shaded leaf node denotes a receiver which also functions as a representative who reports packet survival statistics for its subnet back to the multicast source. Only at the links between (R) and (S) nodes where multicast traffic merges with local terrestrial traffic do we implement one of the following queueing policies: DropTail(FIFO), RED, DRR and DRR+; all other links use DropTail queues. For DRR and DRR+ queues, we can adjust the parameter w_m which is the multiple of 'quantum' the multicast flow receives per service round with respect to that of a unicast flow. In the results shown below, we set $w_m = 4$ which means that, when a service turn arrives, our active multicast flow receives four times of the service quantum of any other active unicast flow receives. The rest of the simulation parameters are shown in Table I.

Parameters	**Values, Notes**
Satellite Channel	$C = 512$Kbps, delay=270ms
(E)-(R) links	512Kbps, 5ms
(R)-(S) links	bandwidths as in Fig.4, 5ms
(I)-(R) links	40Mbps, 20ms
(S)-(Receiver) links	10Mbps, 1ms
Bulk Files	Ten 5-MB files, each encoded in $W = 160$ blocks, $h = 32$ packets
Packet Size	1K bytes (both multicast and TCP)
DRR/DRR+	unicast quantum 1K bytes

Table I: Parameters for the Experiments.

The multicast source can transmit at rate $r \in \mathcal{R} = \{512, 256, 128, 64\}$ Kbps. All the background TCP traffic starts at 0.1 seconds (virtual time). After 30 seconds, the multicast source starts transmitting 2000 probe packets at each and every $r \in \mathcal{R}$ to initialize the packet survival probability estimators (the probe traffic is needed only once at startup, and in practice it can be part of the bulk file). After the probe transmission has ended (and subsequently after each bulk file transmission has ended) all the receiver representatives reliably report their packet survival statistics back to the multicast source via the terrestrial feedback channels. Upon the reception of all the packet survival statistics, the multicast source computes the rate schedule for the next file transmission which starts 5 minutes after the previous file transmission has ended – with an exception of the first bulk file which starts at 600 seconds into the simulation. The rate schedules are computed in such a way that theoretically (i.e. with both one-dimensional independent packet survival and time-invariant assumptions), the probability that each rate schedule provides unreliable delivery of any bulk file to a specific receiver would be at most 0.001 (in reality, these two assumptions are violated, for example by burst losses, thus we may see larger unreliability).

We consider the **bandwidth-delay trade off** in Figs. 5 through 8. On the top of each of these plots and at each of the knob ϕ settings we used in the experiments, we present the **number of globally completed files** which is the number of bulk files that were received completely at **all** of the receivers as a result of actuating the rate schedule alone (i.e. without utilizing any NACK mechanism). In the middle of these plots, and at each knob settings, we present the average **redundancy ratio**, which is defined as $(h + c)/h$ where c is the amount of master block's FEC parity packets, as the result of the rate schedule computation, along with the sample standard deviation from the ten bulk file transmissions. Since we use $h = 32$, if $(h + c)/h < 8$

then the multicast source can use a RSE encoder based on the finite field GF(256), i.e. having 8-bit symbols, which is very common and fast. At the bottom of these plots, and at each knob setting, we show the average **file reception latency** of the files that are globally completed. The file reception latency at a specific receiver is defined as the elapsed time from the moment the receiver sees the start of a bulk file until the receiver receives sufficient packets (data or parity) to reconstruct the whole bulk file.

We also consider the **throughput dynamics** of the multicast and TCP traffic resulting from the queueing algorithms chosen for the (R)-(S) links (Figs. 9 through 12). We choose to see the throughputs during the transmission of the 7th bulk file at knob setting $\phi = 0.1$ (i.e. emphasizing latency reduction rather than bandwidth reduction). In each of these figures, the 'raw' multicast and TCP throughputs as seen from observers Obs #1, #3, and #6 are shown on the top, middle and bottom plots respectively; each data point represents the average throughput in a 5-second interval immediately prior to and excluding the point.

Figure 5: DropTail

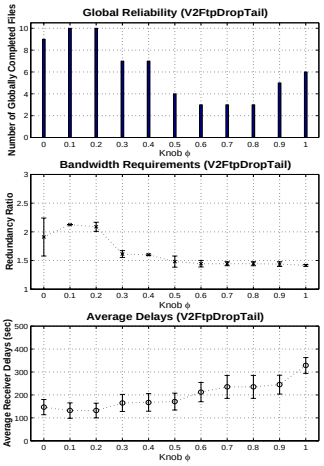


Figure 6: RED

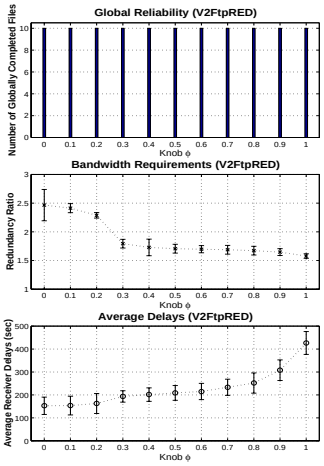


Figure 7: DRR (w_m=4)

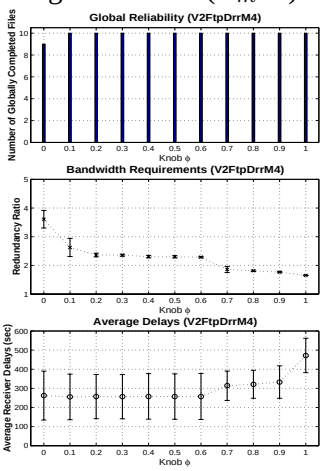
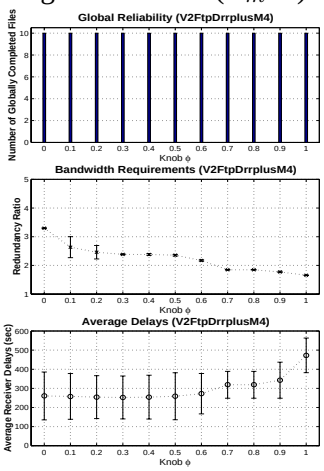
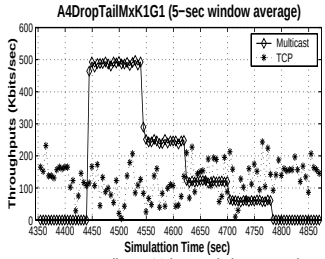


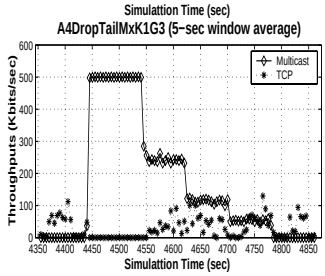
Figure 8: DRR+ (w_m=4)



We now describe our findings. First we notice an effective bandwidth-delay trade off in most of the results we show in Figs. 5 through 8. However, in the use of DropTail queues at the traffic merging points, the global reliability is low (Fig.5) at knob settings around $\phi = 0.5 - 0.9$ (i.e. trying to minimize multicast

Figure 9: DropTail with Knob $\phi = 0.1$





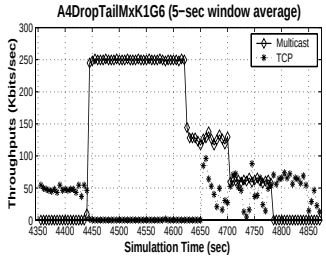
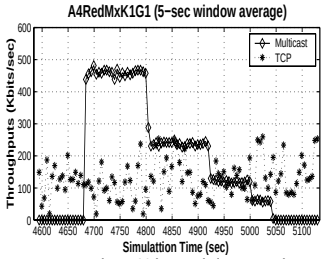
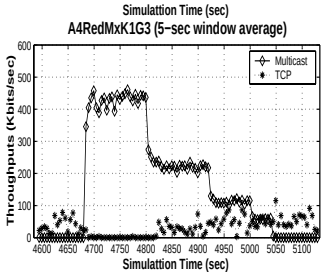
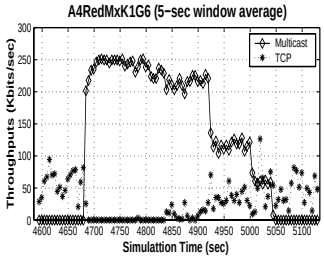


Figure 10: RED with Knob $\phi = 0.1$







bandwidth consumption rather than minimizing the reception latency). The main reason is that burst losses are more common in DropTail queues and thus the receivers connecting to the slowest (R)-(S) link (i.e. the 256Kbps link) sometimes do not receive sufficient packets for some of the $W = 160$ data blocks. The use of RED, DRR, and DRR+ queues at the traffic merging points help in coping with this unreliability problem in different ways. The RED queues helps reduce burst losses by probablistically dropping the packets. DRR has a mechanism to reserve a certain amount of bandwidth for a specific flow; the bursty effect of TCP flows is thus mostly blocked from interfering with the multicast flow and vise versa. DRR+ combines the benefits of both DRR and RED.

Next, we look at the impact of different queueing algorithms on the traffic throughput dynamics. The use of the DropTail queues at the traffic merging points cannot provide any protection for the TCP flow from being bandwidth-starved by the multicast flow, especially at observers (Obs) #3 and #6 (i.e. the middle and bottom plots of Fig. 9 respectively). When compared to the DropTail queues, the RED queues allow a slightly smaller average amount of multicast bandwidth to be passed on to multicast receivers (Fig. 10); this is due to the probabilistic nature of the packet drops. The use of RED queues at the traffic merging points also allows slightly quicker recovery of the TCP throughput at the slowest (R)-(S) links or Obs #6, i.e. comparing the bottom plots of Fig. 9 and Fig. 10. The former shows that it is not until the multicast traffic is being transmitted at rate 128Kbps by the multicast source that the TCP traffic begins to recover, while the latter shows that TCP traffic starts to recover 'sooner' when the multicast traffic is being transmitted at rate 256Kbps. However, the use of RED queues does not completely solve the TCP starvation problem. The use of the DRR queues at the traffic merging point gives better protection (Fig. 11) for TCP. For the 'fastest'

Figure 11: DRR with ϕ=0.1, w_m=4

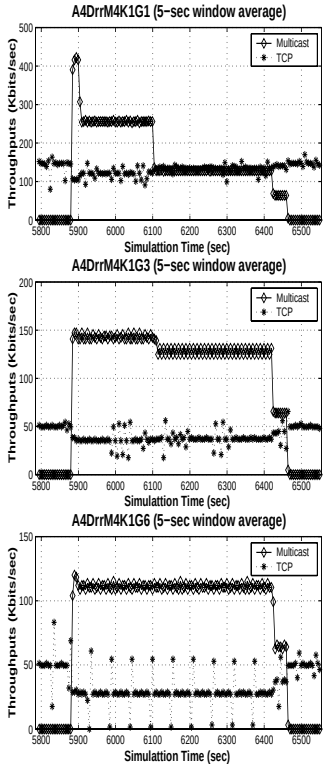
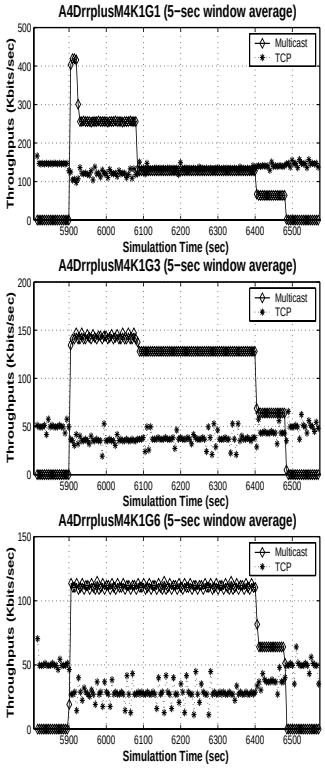


Figure 12: DRR+ with ϕ=0.1, w_m=4



(R)-(S) links (i.e. at observer Obs #1), DRR even gives a 'smoother' TCP throughput than the DropTail's counter part (i.e. the topmost plot of Fig. 11 vs. the topmost plot of Fig. 9). However at the slowest link (i.e. at observer Obs #6, or the bottom plot of Fig. 11), we still notice a recurring pattern of short TCP starvations and restarts. DRR+ almost completely eliminates this problem (Fig. 12). In fact, the DRR+ algorithm, like the RED algorithm, provides an operating environment which is closet to the independent packet survival assumption we used for the rate scheduling mechanism.

We also experimented with different w_m settings for DRR/DRR+ queueing algorithms. If $w_m = 1$, the multicast flow receives no more bandwidth share than any other unicast flow sharing the same (R)-(S) link. In the topology that we use, however, we found that the rate scheduling algorithm is less effective in providing bandwidth-delay trade off because most of the multicast receivers do not benefit from the multicast transmission rates 512Kbps and 256Kbps. This suggests that the choice of rates in the set $\mathcal{R}$ is also important. A solution to this problem can be realized by either using an entirely different set of multicast transmission rates $\mathcal{R}$ or by using some dynamic mechanisms to re-adjust the members of $\mathcal{R}$. The multicast source can generally learn if the rate member in the set $\mathcal{R}$ is appropriate or not by looking at the probed packet survival probabilities $q_r^{(m)}$ at each rate $r \in \mathcal{R}$ and at every receiver $m \in \mathcal{M}$.

Lastly, for DRR/DRR+ queueing algorithms, it is also possible that one can administratively set different w_m values for different subnets according to some local fairness measures, but we do not cover this issue in this paper.

5 CONCLUSION

We have proposed a framework for the rate scheduling of bulk data multicast in a hybrid satellite-terrestrial network. The framework consists of a long-term rate control mechanism that affects a bandwidth-delay trade off and uses a fair queueing algorithm at the points where multicast traffic from the satellite merges with local terrestrial traffic. Among the queueing algorithms we studied, the DRR+ queueing, which was first proposed in [HMMM00], is the most preferable choice because of 1) its compatibility with reno TCP, 2) its capability to provide some fairness to both the satellite multicast traffic and the terrestrial reno TCP traffic, and 3) its conformance to the independent packet survival assumption used in the rate scheduling algorithm.

There is a variety of applications where a satellite bulk data multicast system that can provide a bandwidth-delay trade off is useful. For example, the dissemination of weather information, or (near-) real-time stock prices, and application files. Different file types generally require different degrees of urgency (i.e. the users' satisfaction issue) and network load (i.e. the administrative issue).

Acknowledgements

The authors would like to thank Go Hasegawa of the Osaka University in Japan for the discussion on and the sample codes of the DRR+ queueing. The authors also thank Joseph Macker (NRL) and R. B. Adamson for the discussion on the Multicast Dissemination Protocol (MDP).

References

[BKTN98] S. Bhattacharyya, J. Kurose, D. Towsley, and R. Nagarajan. Efficient rate-controlled bulk data transfer using multiple multicast groups. In *IEEE INFOCOM '98*, March 1998.

[CD99] A. Clerget and W. Dabbous. Tag-based fair bandwidth sharing for responsive and unresponsive flows. Technical Report ISSN 0249-6399 ISRN INRIA/RR 3846, INRIA Sophia Antipolis, France, 1999.

[DAZ99] M.J. Donahoo, M.H. Ammar, and E.W. Zegura. Multiple-channel multicast scheduling for scalable bulk-data transport. In *IEEE INFOCOM '99*, March 1999.

[GB00] M. Grossglaser and J-C Bolot. On service models for multicast transmission in heterogeneous environments. In *IEEE INFOCOM '2000*, March 2000.

[HMMM00] G. Hasegawa, T. Matsuo, M. Murata, and H. Miyahara. Comparisions of packet scheduling algorithms for fair service among connections on the internet. In *IEEE INFOCOM '2000*, March 2000.

[Mac97] J.P. Macker. Reliable multicast transport and integrated erasure-based forward error correction. In *IEEE MILCOM '97*, 1997.

[RJL+00] I. Rhee, S.R. Joshi, M. Lee, S. Muthukrishnan, and V. Ozdemir. Layered multicast recovery. In *IEEE INFOCOM '2000*, March 2000.

[SSZ98] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *ACM SIGCOMM '98*, September 1998.

[ST00] S. Sarkar and L. Tassiulas. Fair allocation of discrete bandwidth layers in multicast networks. In *IEEE INFOCOM '2000*, March 2000.

[SV95] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. In *ACM SIGCOMM '95*, 1995.

[TC00] Apinun Tunpan and M. Scott Corson. Bulk data multicast rate scheduling for hybrid heterogeneous satellite-terrestrial networks. In *IEEE ISCC '2000*, July 2000.

[VRC98] L. Vicisano, L. Rizzo, and J. Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *IEEE INFOCOM '98*, 1998.