A Simple and Scalable Fair Bandwidth Sharing Mechanism for Multicast Flows

Fethi Filali and Walid Dabbous
INRIA, Planète Research Group
2004 Route des Lucioles, BP-93
06902 Sophia-Antipolis, France
Email: {filali,dabbous}@sophia.inria.fr

Abstract

Despite a decade of research and development, multicast has not yet been deployed on a global scale. Among the difficulties with the current infrastructure are inter-multicast fairness support, multicast congestion control, and multicast routing.

In this paper, we focus on the inter-multicast fairness issue which addresses the way how the network resources are shared between competing multicast flows. We propose a simple and scalable single FIFO queue-based active queue management mechanism called MFQ (Multicast Fair Queuing) to achieve the desired inter-multicast fairness. MFQ interacts with an external multicast bandwidth allocation module which implements a pre-defined inter-multicast fairness function. To guarantee a fine-grained packet queuing/dropping, MFQ uses a novel bandwidth sharing notion, called Multicast Allocation Layer (MAL). Based on this notion, MFQ classifies multicast packets into layers and adjusts their weights in order to provide a bandwidth sharing being as close as possible to that given by the fluid model algorithm.

Simulation results demonstrate that MFQ achieves the expected allocation for both responsive and non-responsive multicast flows. We validate our findings by analyzing the impact of network and groups dynamics on the expected bandwidth allocation and comparing it to that obtained by MFQ. Furthermore, we show that MFQ converges very fast to a stable state and easily adapts itself to the dynamic change of the flows weights.

1 Introduction

The IETF orchestrated a very strong guideline for developing a TCP-friendly multicast congestion control scheme [19] regardless of the number of group members. It is well-known that multiplicative decrease/linear increase congestion control mechanisms, and in particular TCP, lead to proportional fairness [15]. However, in [5] the authors have proven that if we treat the multicast flow as if it were a TCP flow, then the application of Kelley, Maullo and Tan's model [15] shows that the larger the multicast group the smaller its share of the proportional bandwidth would be.

We argue that the definition of the inter-multicast fairness (fairness between multicast flows) should take into account the number of competing groups, the number of flows per group, and the number of receivers per flow [22]. In [16] W. Biersack et al. have defined three different bandwidth allocation strategies for multicast flows as well as criteria to compare these strategies. They showed that the LogRD policy[1] always leads to the best tradeoff between receiver satisfaction and inter-multicast fairness. To implement their proposal in real networks, they recommended to introduce their allocation scheme to the general scheduler by configuring the weights of a GPS scheduler [21]. The goal can also be met by reserving the bandwidth in the network for either individual connections or group of connections, and explicitly allocating network bandwidth on a packet-by-packet basis by scheduling packets across network links. However, the two methods are complex because the former requires the use of Fair Queuing mechanisms [8, 21] in each router and the later the use of RSVP-like bandwidth reservation signaling protocols which needs a close coordination and integration between all routers (and hence all network providers) along the path from sender to receiver.

In this paper, we are investigating an alternate approach based on Active Queue Management (AQM) rather than packet scheduling or explicit bandwidth reservation. We

[1] Assume n active multicast flows and denote by n_i the number of downstream receivers of flow i. The receiver-independent (RI), linear (LIN), logarithm (LOG or LogRD) bandwidth sharing policies consist to give to flow i a bandwidth share equal to $\frac{1}{n}$, $\frac{n_i}{\sum_j n_j)}$, and $\frac{1+\log n_i}{\sum_j (1+\log n_j)}$, respectively.

Proceedings of the 10 th IEEE International Conference on Network Protocols (ICNP'02)
1092-1648/02 $17.00 © 2002 IEEE

developed a new active queue management mechanism for routers that (1) provides the expected bandwidth allocation between multicast flows, (2) adapts to the change in multicast group sizes, in the number of active flows, and in the bandwidth allocation strategy used, (3) increases the link utilization ratio. Since social, economic, and technical issues lead the ISPs to implement different fairness policies, considering their business strategy, we made the choice that our mechanism will be independent of the fairness function used. Indeed, the multicast bandwidth allocation module may implement either a multicast fairness function as those described in [16] or a multicast pricing model [6, 12].

To the best of our knowledge there is no prior work on inter-multicast bandwidth sharing using an active queue management mechanism in the open literature. In addition, we consider our scheme a promising avenue of development for congestion control of multicast traffic, and so an additional motivation for this work is to lay a sound basis for further development of multicast congestion control with a help form the network. In absence of similar approaches, MFQ performance is compared to the theoretical expected results.

We call our AQM mechanism, Multicast Fair Queuing (MFQ), a per-flow dropping mechanism which interacts with a bandwidth allocation module that provides for each multicast flow its expected bandwidth allocation[2]. MFQ belongs to the class of per-flow dropping mechanisms like FRED (Flow Random Early Drop) [17]. The operations done by MFQ are not as complex as manipulation of priority queues like FQ (Fair Queuing) [8], given that they only consist of dropping or queuing the packet.

It is important to note that we usually associate the flow-based mechanisms support with complexity and scalability problems since they require connection specific information. These concerns are justifiable only in point-to-point connections, for which routing tables do not maintain connection-specific state. In multicasting, routing tables keep connection specific state in routers anyway; namely, the multicast group address refers to a connection. Thus, adding multicast flow specific information should slightly increase the routing state. We demonstrate that a modest amount of state and computation at network routers can yield significant performance gains for multicast applications while keeping MFQ very **scalable**.

MFQ achieves the expected share of the link bandwidth among all the competing flows using a single FIFO queue. It is designed to be independent of the bandwidth allocation policy. Our mechanism requires routers to maintain state and perform operations on a per flow basis. It uses *Multicast Allocation Layer* (MAL), a novel bandwidth allocation notion that we propose to achieve the expected allocation via a packet-per-packet queuing/dropping manner and to guarantee a very fine grain bandwidth sharing.

We use simulation to evaluate the effectiveness and performance of MFQ for different communication scenarios. We first consider non-responsive multicast source; i.e., sources that do not modify their behavior when a packet loss occurs. Then, we examine the case when the sources implement the Fair Layered Increase/Decrease with Dynamic Layering (FLID-DL) [3] as multicast congestion control mechanism. Finally, we experiment MFQ for heterogeneous multicast flows where there are both responsive and non-responsive competing sources. For the three cases, we will show that the bandwidth share obtained using MFQ is very close to the expected allocation and we will demonstrate that for layered multicast transmission, layers with lower number of members (lower priority layers) will see a loss rate higher than those with higher number of members (higher priority layers).

Without loss of generality, we validate MFQ for both linear and logarithm inter-multicast fairness functions. In addition, we demonstrate that our mechanism adapts itself when there is a big dynamic change in the multicast group size due to join and leave events.

The remainder of this paper is organized as follows. Section 2 details the MAL scheme and the main components of MFQ. We show the simulation results for both responsive and non-responsive multicast sources for both single and multiple congested bottleneck link in Section 3. Section 4 concludes this paper by summarizing results and outlining future work.

2 MFQ: Multicast Fair Queuing

In this section, we present MFQ architecture in two parts. First, we develop the fluid model algorithm. Second, we explore the MFQ components that extend the fluid model algorithm to real networks where transmission is packetized.

2.1 Fluid model algorithm

Before embarking into the mechanism details, let us present the formal network model used to examine our buffer management mechanism.

We consider a network as a set of links $\Im$ where each link l_j has a capacity $C_j > 0$. We assume a known number of multicast flows with different number of receivers. These flows compete for access to communication links. A multicast session S_i is a tuple $(X_i, \{r_{i,1}, ..., r_{i,k}\})$ of session members: X_i is the session sender that transmits data within a network; each $r_{i,p}$ is a receiver that receives data from X_i. Each session contains exactly one sender and

[2] In [10] we have proposed and evaluated a simple scheduler to share the bandwidth fairly between unicast and multicast flows. MFQ was used in the multicast queue.

at least one receiver. We denote G_j the multicast group number j to be a multicast group to which belong one or more multicast sessions. We write $r_{i,p} \in S_i$ to indicate that receiver $r_{i,p}$ is a member of the multicast session S_i and $S_i \in G_j$ to indicate that the multicast session S_i belongs to group G_j.

We define $N_{i,j}$ to be the number of receivers in session S_i whose path toward the source includes link l_j and define N_j to be the number of all receivers whose multicast delivery tree includes link l_j, i.e., $N_j = \sum_i N_{i,j}$.

Considering a single link from the set $\Im$ with a capacity equal to C. We assume m active multicast flows, and that the source number i sends at the instantaneous rate $\alpha_i(t)$, in bits per second.

Usually for unicast flows, we use the unicast max-min fairness [2] to develop the fluid model. For multicast flows, there is no well-accepted definition of the fairness between competing multicast flows. To be independent of the bandwidth allocation strategy used, we assume a vector $\lambda(t) = (\lambda_1(t), \lambda_2(t), ..., \lambda_n(t))$ that defines the expected allocation at the instantaneous time t. The value of $\lambda_i(t)$ may be either a function of the number of competing multicast groups, the number of flows per group, and the number of receivers per flow, or even a set value. We call this vector of fair sharing, the *multicast allocation vector*. If the fair share is achieved, the multicast flow number i receives service at a rate given by $min(\alpha_i(t), \lambda_i(t))$. Let $A(t)$ denote the total arrival rate at the considered link: $A(t) = \sum_{i=1}^{i=n} \alpha_i(t)$. If $A(t) > C$ the congestion phenomena holds and then the multicast allocation vector $\lambda(t)$ is the unique solution to

$$C = \sum_{i=1}^{i=m} min(\alpha_i(t), \lambda_i(t)). \quad (1)$$

If $\alpha_i(t) > \lambda_i(t)$, then the fraction of bits $\frac{\alpha_i(t)-\lambda_i(t)}{\alpha_i(t)}$ will be dropped, and the multicast flow i will have an output rate of exactly $\lambda_i(t)$. The arrival rate to the next hop is given by $min(\alpha_i(t), \lambda_i(t))$.

The number of downstream receivers in each path of the multicast delivery tree does not remain constant because some receivers may be reached via other interfaces other than that belongs to this path. Therefore, the *multicast allocation vector* $\lambda(t)$ may be different in the tree branches even when there is no more competing multicast flows.

2.2 MFQ architecture

Two modules compose our mechanism:

- The multicast bandwidth allocation module which computes the expected fair share for each active multicast flow.
- The buffer management module which uses a single FIFO queue and interacts with the first module to decide the drop preference in order to achieve the expected bandwidth sharing.

We detail in the following sections these two modules.

2.2.1 Multicast bandwidth allocation module

We first present a general framework for the multicast fairness notion. Then, we enumerate some inter-multicast fairness candidates functions that could be implemented by the ISPs. We consider the two existing multicast service models: the ASM (Any Source Multicast) [7] model and the SSM (Source-Specific Multicast) model [13].

2.2.1.1 General framework The multicast bandwidth allocation module determines the link capacity fraction[3] that should be allocated to the flow to which the incoming packet belongs. We develop hereafter a general framework of the multicast bandwidth allocation module. As we have pointed out earlier, the multicast fairness function may depend on the number of groups, the number of flows per group, and the number of receivers per flow. We assume that each ISP has a single and clearly defined multicast bandwidth sharing policy. This policy can be configured in all or some routers inside its network.

Using the network model introduced in Section 2.1, we define the bandwidth γ_{ij}, in bits per second, allocated to group G_i, in link l_j as follows:

$$\gamma_{ij} = F_1(G_i) * C_j \quad (2)$$

where $F_1(.)$ is the ***inter-group multicast fairness function*** that implements the bandwidth sharing policy among active multicast groups. In other words, $F_1(G_i) * C_j$ is the maximum bandwidth, in bits per second, that should be allocated to group G_i in link l_j.

In ASM service model, a multicast group may have one or more sessions (flows) from different sources that share the same communication link. The bandwidth allocated to flow k of the multicast session $S_k \in G_i$ in link l_j is given by the following expression:

$$\lambda_{kj} = F_2(S_p / S_p \in G_i) * \gamma_{ij} \quad (3)$$

where γ_{ij} is computed using Eq. 2. The function $F_2(.)$ determines the fraction of the bandwidth which has already been allocated to group G_i and that should be given to the session S_k. We call this function the ***intra-group multicast fairness function***. This function depends on the number

[3] Throughout this paper we use the terms "link capacity fraction", "flow weights", and "flow bandwidth allocation", interchangeably.

of downstream receivers of each multicast session that belongs to the same group. The functions $F_1(.)$ and $F_2(.)$ must satisfy the following properties:

- for each multicast group G_i: $0 < F_1(G_i) < 1$, and $0 < F_2(S_p) < 1$ for each $S_p \in G_i$.
- in each link l_j: $\sum_i F_1(G_i) = 1$, and $\sum_{pS_p \in G_i} F_2(S_p) = 1$ for each group G_i.

Our main focus in this paper is not to find the "optimal" functions $F_1(.)$ and $F_2(.)$, however we will show that MFQ can adapt itself according to the bandwidth allocation function used, the number of receivers per flow, the number of active flows, and the number of active multicast groups.

2.2.1.2 Examples of inter-multicast fairness functions

In ASM service model, it is possible to have two different sources sending to the same group. It is then sensitive to use the function F_1 to share the bandwidth between multicast groups. Let's assume that the capacity C_j of link l_j is **equitably** shared between them, then the group G_i gets a bandwidth share γ_{ij} equals to $F_1(G_i) * C_j = \frac{1}{g} * C_j$, where g is the number of groups. If F_2 is a logarithm function of the number of receivers, the session $S_k \in G_i$ will get a bandwidth share equal to $\lambda_{kj} = \frac{1+\log n_k}{\sum_{p=1}^{p=m_i}(1+\log n_p)} * \frac{1}{g} * C_j$, where m_i is the number of sessions (flows) that belong to group G_i, and n_p is the number of receivers of session S_p.

In SSM service model, each sender may use a different group address and the multicast session is identified by the couple (sender address, group address). Thus, there is only one source per multicast group and there is no need of using function F_1. In the case of using a logarithm bandwidth sharing function, the session S_k will get a bandwidth share equal to $\lambda_{kj} = \frac{1+\log n_k}{\sum_{p=1}^{p=n}(1+\log n_p)} * C_j$, where n is the total number of competing sessions.

2.2.2 The Multicast Allocation Layer (MAL) scheme

In order to achieve a fine-grained queuing/dropping, we introduce a new scheme, called Multicast Allocation Layer (MAL) which is a key component of the buffer management module. We define a MAL as follows:

Definition: *A MAL is a set of flows that may have the same or different expected allocation in term of the link capacity fraction (the bit-level fairness), but they have the same allocation in term of the maximum number of packets (the packet-level fairness) allowed to be present at the same time in the queue.*

We assume that at the time t, there are n active multicast flows[4] in the queue and that the flow f_i has a weight equal to w_i which is provided by the bandwidth allocation module. We define the MAL mapping function F_{MAL} as follows:

$$\begin{array}{ccc} \{f_1, f_2, ..., f_j, ..., f_n\} & -\!\!\rightarrow & \{0, 1, 2, ..., qlim\} \\ f_j & \vdash\!\!\rightarrow & F_{MAL}(f_j) = \lfloor w_j * qlim \rfloor \end{array}$$

where $qlim$ is the queue size in packets. Two flows f_i and f_j belong to the same MAL number k only if $F_{MAL}(f_i) = F_{MAL}(f_j) = k$.

Given that the number of active flows change, the flows weights and the set of flows per MAL are dynamic. A MAL which has a non-empty set of flows is considered active. We can have at most $qlim$ active MALs in a queue of a size equal to $qlim$ and in this case each MAL contains only one flow.

Let us explain the MAL scheme through the example given in Figure 1. In the x-axis, we show the distribution of 21 active flows in the different MALs. As we can see, there are 5 active MALs with various size in term of the number of active flows that contain each MAL. The flows belonging to the same MAL have different weights drawn in the y-axis by vertical arrows. For example, flows f_{20} and f_3 belong to the MAL number 4 which has 5 active flows $\{f_4, f_{20}, f_{19}, f_3, f_{16}\}$.

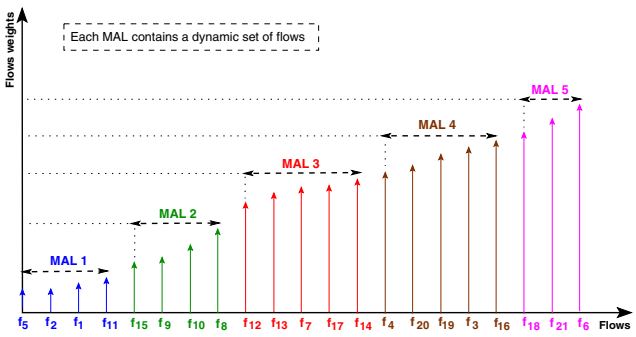


Figure 1. An illustration of the Multicast Allocation Layer (MAL) scheme.

To do a fine-grained queuing, MFQ maintains for each active MAL the identity of the flow which has the highest weight among other flows in the same MAL. As we will detail in the next section, MFQ discriminates between flows belonging to the same MAL in order to achieve the expected fair share.

One can make the observation that when the flows weights are equal, the bandwidth is equitably distributed among flows given that there is only one active MAL including all active flows. We believe that the use of the MAL scheme will be helpful even for unicast flows in differentiated networks by providing a fine-grained queuing when the flow weights are same or different.

[4] A multicast flow is considered instantaneously active if it has at least one packet in the queue.

2.2.3 Buffer management module

The role of the buffer management module is to make the queuing/dropping decision of each incoming multicast packet. In the MFQ design phase, we have taken some key decisions in order to have a suitable mechanism being independent of the network and sources characteristics and especially the variation of flows weights[5], source behavior when a packet is lost, and sources rates.

We use a single FIFO queue with a pre-configured maximum size in packets. For each multicast active flow, we maintain a flow state containing:

- the number of packets belonging to this flow which are waiting to be served,
- the current flow weights which is provided by the bandwidth allocation module.

Given that we do queuing using per-packet manner and not per-bit manner, MFQ tries to guarantee that the maximum number of packets allocated to each active flow i remains always less than the integer value of its expected allocation **in term of the capacity fraction** w_i multiplied by $qlim$, the maximum queue size in packets, i.e.; $\lfloor w_i * qlim \rfloor$. However, two flows that have different weights may have the same maximum number of packets allowed to be queued. In the example given in Figure 1, the maximum number of allowed packets of flow number 3 and flow number 20 is equal to 4 because $\lfloor w_3 * qlim \rfloor = \lfloor w_{20} * qlim \rfloor = 4$ where $w_3 = 0.07$, $w_{20} = 0.065$ and $qlim = 64$. To guarantee a more fine-grained queuing we introduced in Section 2.2.2 the MAL scheme which we will use in the buffer management module.

For every arriving packet, the router starts by identifying the multicast flow and its MAL.If the flow is new or if there is a change on the number of receivers[6], we get the new flow weight from the bandwidth allocation module.

Let flow number i be the flow to which belong the arriving packet and j be the number of its MAL. We generate a random value $u \in [0, 1]$ and we allow the flow i gets **one more packet** than its MAL if $u < w_i / MAL[j].maxAllocation$, where $MAL[j].maxAllocation$ is the maximum weight of flows belonging to the MAL number j. As consequence, we ensure that each two flows that belong to the same MAL will get **randomly and proportionally** to their fair share one more packet than their MAL and we ensure a much more fine-grained bandwidth sharing.

To prevent the queue from being monopolized by high-rate or bursty multicast sources, we use a pre-configured threshold variable ***thrsh***. If the mean queue size[7] is less than *thrsh* the packet will be accepted only if the number of waiting packets belonging to the flow does not exceed the allowed number (its MAL or its MAL plus one more packet depending on the generated number *u* as explained above).

If the mean queue size is more than the threshold *thrsh* or the queue is full, we accept the packet only if it belongs to an inactive flow. If the queue is full, we drop the incoming packet if its flow is active, otherwise we drop randomly a packet from the queue and we queue the incoming packet. By this way we allow a new multicast flow to become active and we remove the bias against bursty sources. If the packet was accepted, we update the flow and the MAL state.

2.3 Complexity and implementation issues

At each MFQ router, we need to maintain a state per active multicast flow. Upon a packet arrival, the router needs to (1) determine the flow and the MAL number to which the arriving packet belongs, (2) update the flow state and the MAL state parameters such as the number of packets of the corresponding flow and the allocation of this flow provided by the multicast bandwidth sharing module. As shown in [23], these operations and even the packet classification could be efficiently implemented because it only consists of reading the flow ID for IPv6 or the pair (source IP address, multicast destination address) for IPv4.

At branch points in the multicast tree, MFQ routers must record additional information needed by the multicast bandwidth allocation module such as the number of downstream receivers. The processing complexity at routers is increased in two (minor) ways. First, during the lookup-operation for each packet arrival, useful information must also be retrieved from the routing table entry. Second, before accepting the datagram, MFQ should verify that the flow is authorized to get more packets in the queue. While our mechanism would benefit from better bandwidth allocation functions, it is explicitly designed to be robust to coarse implementation of the inter-multicast fairness function using the MAL scheme described in Section 2.2.2.

It should also be noted that the source address and the destination group address are not only needed by the MFQ

[5] In the case of using a fairness function which depends on the number of downstream receivers, all flows weights change when at least one receiver joins or leaves one of the multicast active sessions.

[6] We assume that we know the number of downstream receivers for each active multicast flow and in each router belonging to the multicast delivery tree. In [9], we have proposed an extension to the multicast service to allow senders as well as intermediate routers to explicitly and efficiently count the number of downstream members in each outgoing interface.

[7] We use the same method as RED (Random Early Drop) [11] to estimate the mean queue size $qlen$. The formula for calculating the average queue length $qlen$ is $qlen = (1 - W_q) * qlen + W_q$, $0 \leq W_q \leq 1$. The weighted moving average formula, with weight W_q is used to filter out transient congestion. The value of W_q is set to 0.002 in all simulations.

mechanism but also by the multicast routing lookup module which has to determine the list of outgoing interface(s) for each incoming multicast packet.

2.4 Incremental deployment

MFQ does not escape to the rule that any network service must be able to be deployed in an incremental fashion on the Internet, due to the scale and inherent heterogeneity of the network. It allows to enhance performance even if it is supported only by specific few routers. In addition, there is no need to use MFQ in routers that have a low multicast traffic. Then, MFQ can be implemented/activated in some routers that the ISP administrative authority consider to handle an important amount of multicast traffic load.

As a first step in MFQ deployment, it could be implemented/activated in the multicast border routers that implement the inter-domain multicast routing MBGP (Multicast Border Gateway Protocol). Using MFQ, the border routers can provide a diffserv-like service for arriving multicast flows from directly-connected domains based on a predefined bandwidth sharing function.

2.5 MFQ and layered multicast

When evaluating new network control mechanisms one must evaluate the impact of the proposed mechanisms on application performance. We discuss in this section the MFQ impact on layered multicast application performance.

When the multicast session using a layered transmission scheme, the data is split into layers and each layer sends to a different group address. Depending on the loss rate seen by the receivers, they join and leave layers to adapt to the network situations. We demonstrate how MFQ can achieve a priority dropping without explicitly assigning priorities to the transmission layers.

Assuming a multicast source decodes data into n transmission layers and that there are R_i receivers subscribed to the layer l_i (l_0 is the base layer). Given that receivers who join the layer l_i should join all lower layers $l_0 ... l_{i-1}$, we can easily write the following inequality: $R_{n-1} \leq ... \leq R_j \leq R_{j-1} \leq ... \leq R_1 \leq R_0$.

Without loss of generality, we assume the use of the LogRD function to allocate the bandwidth fairly between multicast flows. The weight of the transmission layer number j is $w_j = \frac{1+\ln R_j}{\sum_{p=0}^{p=n}(1+\ln R_p)}$. We can easily write the following inequality: $w_{n-1} \leq ... \leq w_j \leq w_{j-1} \leq ... \leq w_1 \leq w_0$. We can then write $F_{MAL}(f_{n-1}) \leq ... \leq F_{MAL}(f_j) \leq F_{MAL}(f_{j-1}) \leq ... \leq F_{MAL}(f_1) \leq F_{MAL}(f_0)$, where $F_{MAL}(f_i)$ is the MAL to which belongs the flow associated to the transmission layer number i. Thus, it is clear that layers with lower number of receivers (lower priority layers) will see a loss rate higher than those with higher number of receivers and in particular the base layer l_0 (highest priority layer).

Similar approaches that need a network-support including priority dropping [1] schemes require that the network support as many loss priority levels as layers. In addition, to ensure a fair allocation of resources, they also require that each session uses the same set of priorities than others. Furthermore, priority dropping provides no incentives for receivers to lower their subscription level.

3 Simulation methodology and results

We have examined the behavior of MFQ under a variety of conditions. We use an assortment of traffic sources and topologies. All simulations were performed in ns-2 [20], which provides accurate packet-level implementation for various network protocols.

3.1 Non-responsive multicast flows

We start by validating MFQ for a simple topology consisting of a single congested link connecting two routers $n1$ and $n2$ and having a capacity C equal to 10 Mbps and a propagation delay D equal to 1 ms. As shown in Figure 2, the multicast sources are connected to router $n1$ and receivers are downstream to router $n2$.

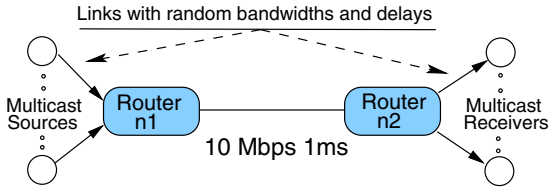
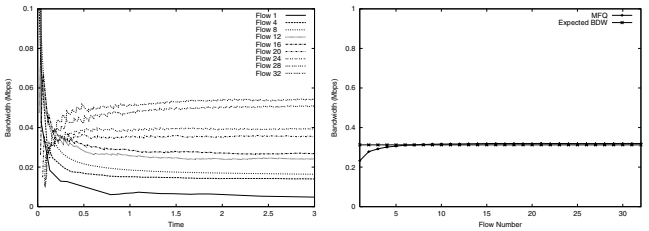


Figure 2. A single congested link.

We compare the expected bandwidth with that obtained by MFQ for 32 multicast flows. The maximum buffer size $qlim$ used in all our simulations is set to 64 packets. We index the multicast flows from 1 to 32 and we compute the bandwidth share of each flow. We use linear and logarithm bandwidth allocation function. When we use a linear allocation function, the fair share of flow i is equal to $\frac{i}{\sum_{j=1}^{j=32} j} * C$ and it is equal to $\frac{1+\lg i}{\sum_{j=1}^{j=32}(1+\lg j)} * C$ in the case of using a logarithm allocation function.

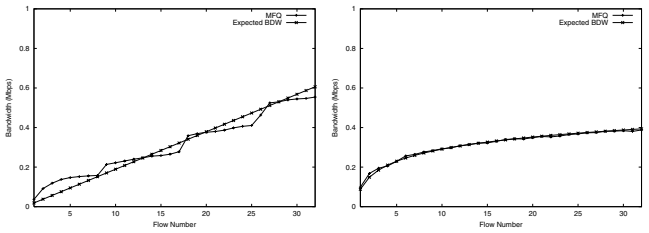
In this section, we assume that each multicast source is a non-responsive CBR source. For this case, we use a CBR generator simulating a non-adaptive audio application. We assume that the source i sends data at a rate equal to i more that its expected fairness rate provided by the fairness function. For example, there is a total amount of $\sum_{i=1}^{i=32} \frac{i}{32} * C = 16.4 * C$ kbps arriving at the bottleneck



(a) Convergence of MFQ

(b) The bandwidth share provided by MFQ when each multicast flow has exactly one receiver

Figure 3. Performance Evaluation of MFQ.



(a) Using a linear multicast allocation function

(b) Using a logarithm multicast allocation function

Figure 4. 32 CBR multicast flows. The flow number i has exactly i downstream receivers.

link when we use a receiver-independent multicast fairness function. Thus, the flow 1 sends 0.3125 *Mbps*, and flow 2 sends 0.625 *Mbps*, and so on.

Unless otherwise specified, each simulation lasts 30 seconds, the source number i starts sending data $i * 0.001$ sec after the simulation starting time and the packet size is assumed to be equal to 1000 bytes. To help the understanding of our result plots and without loss of generality we assume that the flow number i has i downstream receivers.

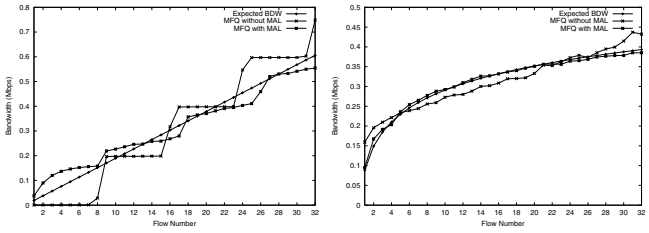
In a first experiment, we focus on the convergence phase of MFQ. We use a linear bandwidth allocation function and we plot in Figure 3(a), the variation of the bandwidth share of some flows in function of the simulation time. We can see that for all flows MFQ reaches a steady sate after approximately one second of simulation. In addition, as expected the flow number i gets more bandwidth share than flows $1, ..., (i-1)$ giving that we use a receiver-dependent fairness function.

In all the following simulations, we start the measurements one second after the simulation starting time so that, the current and the average queue size have already reached a stable state.

In a second experiment, we validate our mechanism for a simple case where each multicast flow has only one receiver which corresponds also to the case when using a receiver independent bandwidth allocation function. Figure 3(b) shows the bandwidth share obtained by MFQ. Regardless the multicast bandwidth allocation function used, all the flows get the same bandwidth share $\frac{1}{32} * C = 0.3125\ Mbps$. Thus, MFQ achieves a good precise degree of fairness between multicast flows.

In Figure 4(a) and Figure 4(b) , we plot the bandwidth share of each multicast flow when using a linear and a logarithm allocation function, respectively. We can see that the bandwidth allocation provided by MFQ is close to the expected fairness for both cases.

The third experiment aims to demonstrate how MFQ performance can be improved by the use of the MAL scheme. In Figure 5(a), and Figure 5(b), we plot the bandwidth share obtained by MFQ with and without the MAL scheme. According to the plots, our MAL scheme can achieve a good fine-grained bandwidth sharing. In addition, we can easily see the distribution of flows in MALs when we do not use the MAL scheme. For example, in the case of linear allocation function (Figure 5(a)), flows from 9 to 15 belong to the same MAL number 2, flows from 16 to 24 belong to the MAL number 3, and so on.



(a) Using a linear multicast allocation function

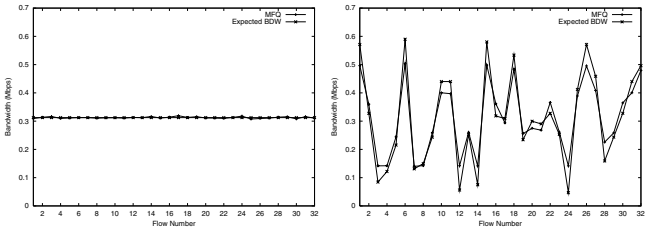
(b) Using a logarithm multicast allocation function

Figure 5. 32 CBR multicast flows.

Without using the MAL scheme, flows 9 and 15 (belong to the MAL number 2) will get the same bandwidth share, which is not fair. The use of MAL allows these

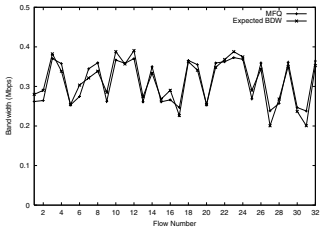
flows to share the bandwidth fairly. Indeed, flow 15 gets more bandwidth share than flow 9.

In the previous experiments we assumed that the number of receivers for each flow is equal to the flow index. Now, we assume that the number of receivers of each flow is randomly generated between 1 and 64. For all other parameters, we use the same values as the second experiment. We show in Figure 6(a), Figure 6(b), and Figure 6(c) the obtained bandwidth for each flow when we use a receiver-independent allocation function, a linear allocation function, and a logarithm allocation function, respectively. These results confirm that our mechanism is independent of the number of receivers per group and it is independent of the multicast fairness policy used. We conducted more simulation experiments where the number of receivers per flow is randomly generated. Our results that may not be presented here due to space limitations match what we expect and they demonstrate the ability of MFQ to adapt to the change on the number of receivers when using a receiver-dependent inter-multicast fairness function.



(a) RI fairness function

(b) LIN fairness function



(c) LOG fairness function

Figure 6. 32 CBR multicast flows. The number of receivers of each flow is randomly generated.

3.2 Responsive multicast flows

In this section, we examine the behavior of MFQ in presence of responsive multicast sources where the senders use the layered transmission scheme which was first proposed for the RLC (Receiver-driven Layered Congestion) congestion control protocol [24]. To address some of the deficiencies of RLC, authors of [3] propose Fair Layered Increase/Decrease with Dynamic Layering (FLID-DL) protocol [18]. The protocol uses a Digital Fountain [4] mechanism at the source. FLID-DL introduces the concept of Dynamic Layering to reduce the join and leave latencies associated with adding or dropping a layer. With Dynamic Layering, the bandwidth consumed by a layer decreases over time. Thus a receiver has to periodically join additional layers to maintain its receive rate. The receive rate is reduced simply by not joining additional layers, whereas rate increase requires joining multiple layers.

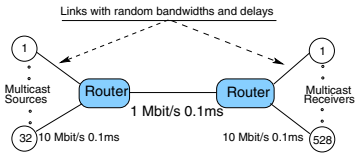
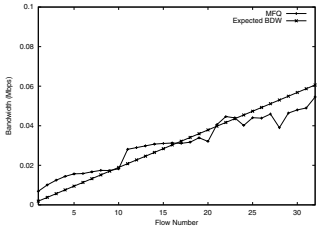
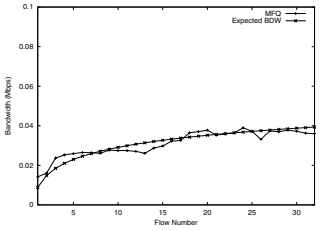


Figure 7. Network configuration.



(a) Using a linear bandwidth allocation function



(b) Using a logarithm bandwidth allocation function

Figure 8. 32 FLID-DL sources.

In this experiment, we assume that we have 32 multicast competing sources that use the FLID-DL congestion control protocol. We assume that the source number i has exactly i receivers which result on 528 different receivers. We use the network configuration given in Figure 7.

Once a receiver has obtained the transmission session description, which includes the information about the groups associated with a session that is needed in order to

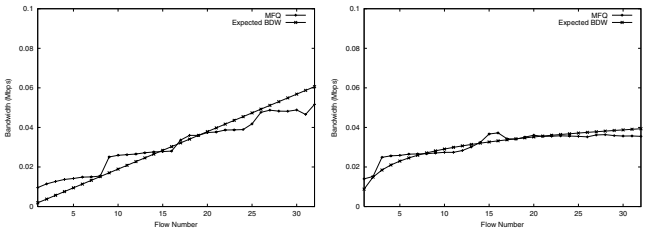
join those groups, it may join one of the groups in the session. The receivers must join and leave groups within a session as described in detail in [18].

Each source uses 17 layers encoding, each layer is modeled by a UDP traffic source. In Figure 7(b), we give the values of FLID-DL parameters set in the FLID-DL ns simulation.

The performance of MFQ in presence of FLID-DL sources is shown in Figure 8(a) and Figure 8(b) for both linear and logarithm bandwidth allocation function, respectively. As can be seen from the plots, MFQ enforces the required fairness very well. The obtained results are slightly different from the case of CBR non-responsive sources because the FLID receivers join and leave layers according to the loss rate observed. That's why we see an oscillation in the obtained plot around the expected plot.

3.3 Heterogeneous multicast flows

We have conducted simulation to evaluate the performance of MFQ when we have heterogeneous multicast sources: responsive and non-responsive. For this end, we use 32 multicast sources indexed from 1 to 32 where flows from 1 to 16 are generated by FLID-DL sources and flows from 17 to 32 correspond to CBR sources.



(a) Using a linear allocation function

(b) Using a logarithm allocation function

Figure 9. 32 multicast flows. Flows form 1 to 16 are FLID-DL sources and those from 17 to 32 are CBR sources.

We use the network configuration of Figure 7(a), and we suppose again that the flow i has exactly i receivers. The CBR sources are similar to those of the first experiment of Section 3.1, and the FLID-DL parameters are similar to those of Section 3.2.

We plot in Figure 9(a) and Figure 9(b) the obtained and the expected bandwidth sharing for linear and logarithm bandwidth allocation policy, respectively. As show in these figures, MFQ matches closely the fair share for both policies despite the heterogeneity of the multicast sources.

3.4 Responsiveness to group size dynamics

An important concern in the design of active queue management mechanisms is their responsiveness to changes in flow weights. To illustrate how MFQ adapts to that change, we assume the use of LogRD fairness function so that the change on the number of receivers affects the expected fair share of all active flows. When receivers join and leave the multicast session, it is important that MFQ reacts sufficiently fast should a change of *multicast allocation vector* be required. This behavior is investigated by randomly generating join and leave events. We measure how long MFQ takes to adapt to the variation of the flow weights.

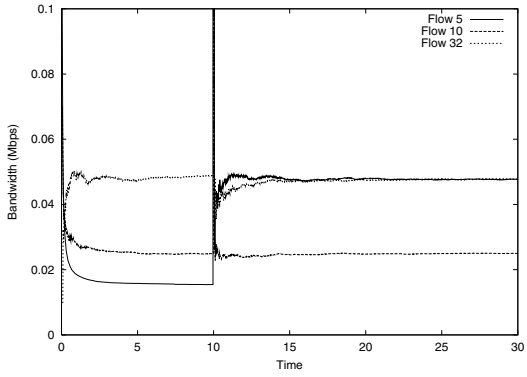


Figure 10. 32 multicast flows using a linear allocation function. At 10 seconds of simulation 27 new receivers join the multicast session number 5.

We consider the single link of Figure 2 and we use the same configuration as the first experiment of Section 3.1. After 10 seconds of simulation we increase the number of receivers of the flow number 5 from 5 to 32 to emulate join events arriving towards the source from $27\ (32-5)$ new receivers.

As shown in Figure 10, MFQ mechanism adapts to the change on the number of receivers of flow number 5. Indeed, we see that after one second of the arriving of the new join events, the flow number 5 gets exactly the same bandwidth as the flow number 32 which has already 32 receivers. We can also see that the flow number 10 has not been affected by this variation only for one second. As

expected, each flow gets a slightly less bandwidth share after increasing the number of receivers of flow number 5. This experiment demonstrates that MFQ reacts rapidly to the change in flow weights.

The same simulation configuration can be used to investigate responsiveness to changes in sizes of many multicast groups. The results (not shown here) are similar to those above, since all flows get their fair share fatly.

3.5 different starting times

In this experiment, we measure how MFQ reacts when the multicast sources have different starting time. Again we consider the single link of Figure 2 and we use the same configuration as the first experiment of Section 3.1. In particular, the flow number i has i downstream receivers, and a logarithm fairness function is used to share the bandwidth between competing flows. We assume that the flow number i starts sending 2 seconds after the flow number $i-1$. The flow number 1 starts at $0.001\ sec$.

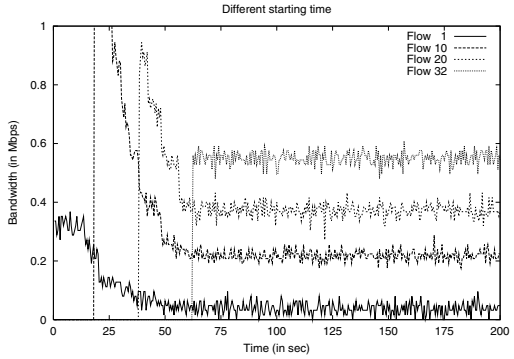


Figure 11. 32 CBR multicast sources using a logarithm allocation function. The flow number i starts 2 seconds after the flow number $i-1$.

In Figure 11, we plot the bandwidth share variation of flows 1, 10, 20, and 32 in function of the simulation time. Two main observations can be derived from the obtained plots. Firstly, when the flow number i starts, the bandwidth share of all flows number $1...(i-1)$ decreases to reach a new stble state. Secondly, the flow number i gets more bandwidth share than all its lower indexed flows (flows from 1 to $i-1$). MFQ achieves the bandwidth sharing according to the logarithm multicast fairness function. Indeed, we can observe that the bandwidth share increases logarithmically with the number of receivers.

3.6 Multiple congested links

In this sub-section, we analyze how the results are affected when the flow traverses L congested link. We index the links from 1 to L and the capacity C_1 of the link number 1 is set to 1 Mbps. A link j, $2 \leq j \leq 10$, is kept congested by setting its capacity C_j to $C_{j-1} - 50$ Kbps.

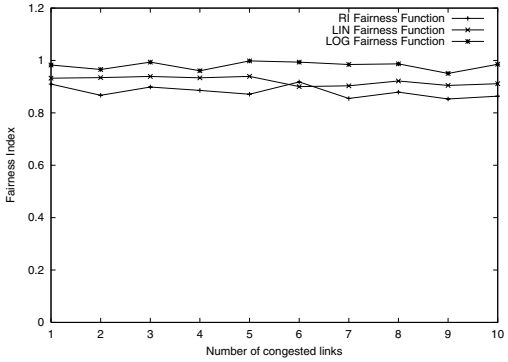


Figure 12. MFQ performance in multiple bottleneck link.

We extended the unicast fairness index introduced in [14] to multicast traffic. Instead of the expected unicast fair rate, which is always the same for unicast connections, we use that of the multicast flow which may differ from one flow to another depending on the multicast bandwidth allocation policy implemented. The **multicast fairness index** is computed as follows:$1 - \frac{1}{n}\sum_{i=1}^{i=n} |\frac{t_i - \overline{t_i}}{\overline{t_i}}|$, where n is the number of active multicast flows, $\overline{t_i}$ and t_i are the expected and the obtained bandwidth share of the multicast flow i, respectively. We plot in Figure 12, the variation of the fairness index as a function of the number of congested links. As we can see, it remains close to 1 even when the number of congested increases for RI, LIN, and LOG fairness functions. As expected in [16], the LOG fairness function has better fairness index variation than the two others functions.

4 Conclusion and future work

In this paper we have presented MFQ, a multicast active queue management mechanism that achieves the expected multicast bandwidth sharing using a single FIFO queue. MFQ interacts with a fairness module which implements the multicast bandwidth allocation policy. The queuing and the dropping decision are designed in a man-

ner to provide a bandwidth sharing being as close as possible to that achieved by the fluid model algorithm. MFQ uses a threshold to penalize high rate multicast sources and accepts packets from new flows to be queued.

Without loss of generality, MFQ is evaluated for both linear and logarithmic bandwidth allocation functions. The scheme is also applied in the presence of both responsive and non-responsive flows. We showed that MFQ performs well even when there is a dynamic change in the number of receivers, since it converges very fast to the new expected bandwidth fair share.

MFQ combined with a well-accepted definition of the inter-multicast fairness provides a significant step towards a complete and scalable congestion control algorithm for multicast applications. We hope that the introduction of MFQ will encourage the ISPs to support the multicast in their networks because it provides a flexible way to share the bandwidth between competing multicast flows.

Future work could evaluate the performance for other types of multicast traffic that include different application-based or transport-based congestion control mechanisms.

References

[1] S. Bajaj, L. Breslau, and S. Shenker, *Uniform versus priority dropping for layered video*, In Proc. of SIGCOMM'98, September 1998

[2] D. Bertsekas and R. Gallagher, *Data Networks*, Englewood Cliffs, NJ, Prentice-Hall, 1992.

[3] J. Byers et al., *FLID-DL: Congestion Control for Layered Multicast*, In Proc. of NGC 2000, November 2000.

[4] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, *A digital fountain approach to reliable distribution of bulk data transfer,* In Proc. of ACM SIGCOMM'98, September 1998.

[5] D. M. Chiu., *Some Observations on Fairness of Bandwidth Sharing*, In Proc. of ISCC'00, July 2000.

[6] L A. DaSilva, *Pricing for QoS-Enabled Networks: A Survey*, In the Journal of IEEE Communications Surveys, pp. 2-8, Second Quarter 2000.

[7] S. Deering, *Host Extensions to IP multicasting*, IETF, RFC 1112, May 1990.

[8] A. Demers, S. Keshav, and S. Shenker, *Analysis and Simulation of a fair queuing algorithm*, Internet working: Research and Experience, V. 1, No. 1, pp. 3-26, 1990.

[9] F. Filali, H. Asaeda, and W. Dabbous, *Counting the Number of Members in Multicast Communication*, In Proc. of NGC 2002, October 2002.

[10] F. Filali and W. Dabbous, *SBQ: A Simple Scheduler for Bandwidth Sharing Between Unicast and Multicast Flows*, In Proc of QofIS 2002, October 2002.

[11] S. Floyd, V. Jacobson, and V. Random, *Early Detection gateways for Congestion Avoidance,* IEEE/ACM TON, V.1 No.4, pp. 397-413, August 1993.

[12] T. N.H. Henderson, *Protocol-independent multicast pricing*, In Proc of NOSSDAV'00, June 2000.

[13] H. Holbrook and B. Cain, *Source-Specific Multicast for IP*, IETF, Internet draft: draft-ietf-ssm-arch-00.txt, May 2002.

[14] R. Jain, *The art of computer systems performance analysis*, John Wiley and sons QA76.9.E94J32, 1991.

[15] F. Kelly, A. Maulloo and D. Tan, *Rate control in communication networks: shadow prices, proportional fairness and stability*, Journal of the Operational Research Society 49, pp. 237-252, 1998.

[16] A. Legout, J. Nonnenmacher, and E. W. Biersack, *Bandwidth Allocation Policies for Unicast and Multicast Flows,* IEEE/ACM TON, V.9 No.4, August 2001.

[17] D. Lin and R. Morris, *Dynamics of Random Early Detection*, In Proc. of ACM SIGCOMM'97, September 1997.

[18] M. Luby, L. Vicisano, and A. Haken, *Reliable Multicast Transport Building Block: Layered Congestion Control*, Internet draft: draft-ietf-rmt-bb-lcc-00.txt, November 2000.

[19] A Mankin, et al., *IETF Criteria for evaluating Reliable Multicast Transport and Applications Protocols*, IETF RFC 2357, June 1998.

[20] S. McCanne and S. Floyd, *Ucb/lbnl/vint network simulator (ns) version 2.1b6,* `http://www-mash.cs.berkeley.edu/ns/`, 2000.

[21] A. Parekh and R. G. Gallager, *A generalized processor sharing approach to flow control - the single node case*, In Proc. of IEEE INFOCOM'92, May 1992.

[22] J. Shapiro, D. Towsley, and J. Kurose, *Optimization-Based Congestion Control for Multicast Communications*, In Proc of IEEE INFOCOM' 2000, March 2000.

[23] D. C. Stephens, J.C.R. Bennet, and H. Zhang, *Implementing scheduling algorithms in high speed networks*, IEEE JSAC, V. 17, No. 6, pp. 1145-1159, June 1999.

[24] L. Vicisano, L. Rizzo, and J. Crowcroft, *TCP-like congestion control for layered multicast data transfer*, In Proc. IEEE INFOCOM, March 1998.