"Do you really need multicast? At line rates?"

— Less is More[†]

J

Parallel Packet Copies for Multicast

J.1 Introduction

Multicasting is the process of simultaneously sending the same data to multiple destinations on the network in the most efficient manner. The Internet supports a number of protocols to enable multicasting. For example, it supports the creation of multicast groups [221], reserves a separate addressing space for multicast addresses [222, 223], supports the ability to inter-operate between multicast routing protocols [224], and specifies the support needed by end hosts [225] in order to provide an efficient multicast infrastructure.

Among the applications that require multicasting are real-time applications such as Telepresence [28], videoconferencing, P2P services, multi-player games, IPTV [29], *etc.* Multicasting is also used as a background service to update large distributed databases and enable coherency among Web caches.

While we cannot predict the future uses and deployments of the Internet, it is likely that routers will be called upon to switch multicast packets passing over very-high-speed lines with guaranteed quality of service. Specific enterprise networks already exist on which multicasting (primarily video multicasting) is expected to exceed 15-20% [226] of all traffic. In such cases, routers will have to support the

[†]Cisco Systems, Bangalore, India, Oct 2007.

extremely high memory bandwidth requirements of multicasting. This is because an arriving multicast packet can be destined to multiple destinations, and the router may have to make multiple copies of the packet.

J.1.1 Goal

In this appendix, we briefly look at how high-speed routers can support multicast traffic. Our goal is to find the speedup required to precisely emulate a multicast OQ router so that we can provide deterministic performance guarantees. We will consider both FCFS and PIFO queueing policies.

J.1.2 Organization

In what follows, we will describe a load balancing algorithm for multicasting, and briefly discuss a solution based on caching. The rest of this appendix is organized as follows: In Section J.2, we describe two orthogonal techniques, copy and fanout multicasting, to support multicast traffic. In Section J.3, we describe a hybrid strategy to implement multicast, which we will later show is more efficient than either copy or fanout multicasting. We then use the constraint set technique in Section J.4 (first described in Chapter 2) to derive a load balancing algorithm to support multicast traffic on single-buffered routers. We use the parallel packet switch (PPS) architecture (introduced in Chapter 6) as an illustrative example of single-buffered routers, and use constraint sets to bound the speedup[1] required to support multicast traffic. Later, we show that a similar load balancing algorithm can be used to find the conditions under which any single-buffered multicast router can give deterministic performance guarantees. In Section J.5 we briefly consider caching algorithms to support multicasting.

J.2 Techniques for Multicasting

A multicast packet (by definition) is destined to multiple different outputs. Thus, any router that supports multicast must (at some point) create copies of the packet. A

[1]Recall that in a PPS, we use internal links that operate at a rate $S(R/k)$, where S is the speedup of the internal link.

number of router architectures have been studied, and many techniques have been proposed to enable these architectures to support multicast. A survey of some of these techniques and the issues involved in supporting multicast traffic can be found in [227, 228, 229].

☞**Observation J.1.** Note that a memory only supports two kinds of operations — a write and a read. The only way to create copies is to use either of these two operations. Therefore, in what follows we will define two orthogonal techniques to create copies of packets, called "copy multicasting" (where a packet is written multiple times) and "fanout multicasting" (where multicasting is done by reading a packet multiple times). Since a memory supports only two operations, all of the multicasting techniques described in [227, 228, 229] use a combination of the above two orthogonal techniques to implement multicasting.

J.2.1 Copy Multicasting

ℵ**Definition J.1. Copy Multicast:** In "copy multicast", multiple unicast copies (one destined to each output of the multicast cell) are created immediately on arrival of the cell. Each copy is stored in a separate unicast queue.

Once the unicast copies of a multicast cell are created, these unicast cells can be read by their respective outputs and released from the buffer independent of each other. Note that in a PPS, the unicast copies must be immediately sent from the demultiplexor to the center stage OQ switches. Similarly, in an SB router, the unicast copies must be immediately written to memories.

☞**Observation J.2.** Note that copy multicasting places a tremendous burden on the bandwidth of a router. A multicast packet arriving on a line card at rate R, with multicast fanout m, requires an instantaneous write bandwidth of mR. It is a surprising fact that in spite of this, copy multicasting is commonly used in high-speed Enterprise routers. This is because buffer resources such as queue sizes, free lists, *etc.*, can be better controlled using copy multicasting. As we shall see shortly, it also does not suffer from what is colloquially called the "slow port" problem. Most high-speed Enterprise routers today are built to support no more than a fanout of two to three at line rates. If a sustained burst of multicast packets with larger fanouts arrives, these multicast packets will initially get buffered on-chip, and then if the on-chip buffer capacity is exceeded, they will eventually be dropped.[2]

J.2.2 Fanout Multicasting

ℵ**Definition J.2. Fanout Multicast:** In "fanout multicast", a multicast cell is stored just once. The cell is read multiple times, as many times as the number of outputs to which it is destined.

In a PPS using fanout multicast, for example, the cell is sent just once to one of the center stage OQ switches. The layer to which the multicast cell is sent must therefore adhere to the output link constraints of all its outputs simultaneously. In a PPS using fanout multicast, the single center stage switch that is chosen to receive the multicast cell is responsible for delivering the cell to every output.[3] Similarly, in an SB router, an arriving multicast cell is written to a memory that simultaneously satisfies the output link constraints of all outputs.

A standard way in which fanout multicasting is implemented on a router is to write the packet once to memory, and create multiple descriptors (one for each of the queues to which the multicast packet is destined) that point to the same packet. As

[2]Routers are usually built to give priority to unicast packets over multicast; so arriving unicast packets are never dropped, as far as possible.

[3]Note that the center stage OQ switch may in fact take the multicast cell and store it in a common shared buffer memory that is accessible to all its outputs, or it may actually make multiple copies of the multicast cell, if the memories for each of its outputs are not shared. How the multicast cell is handled by the center stage OQ switch is of no consequence to our discussion on fanout multicast. The fact that it may itself make local copies should not be confused with PPS copy multicasting. From the demultiplexor's perspective, if the PPS sends the multicast cell just once, then it is performing fanout multicasting.

an example, refer to the descriptor and buffer hierarchy shown in Figure 8.2. Fanout multicasting is easily implemented by making the multiple descriptors point to the same multicast packet.

☞**Observation J.3.** Fanout multicasting suffers from what is colloquially called the *"slow port"* problem. Since a multicast cell is stored only once in the buffer, it can only be released when *all* the ports that participate in the multicast have completed reading the cell. If one of the ports is congested, that port may not read the multicast cell for a very long time. Such multicast cells will hog buffer resources, and if there are many such cells, they can prevent new cells from being buffered.

J.2.3 Comparison of Copy and Fanout Multicasting

A simple way to understand copy and fanout multicasting is that copy multicasting creates copies *immediately* when a packet arrives, while fanout multicasting creates copies *just in time* when a packet needs to be sent to the output (or transferred across the switch fabric). It is clear from the definition that a router requires more write bandwidth in order to support copy multicasting, because the copies have to be made *immediately*. Since packets are still read at the line rate, no read speedup is required. For fanout multicasting, packets are written (and read) at the line rate (because no copies need to be made), and so it would appear that no additional memory bandwidth is required to support fanout multicasting. However, since we are interested in providing deterministic performance guarantees, we will see that this is not the case.

We can make the following observations about multicasting:

☞**Observation J.4.** A multicast PPS router at rate R that copy multicasts with maximum fanout m appears to the demultiplexor as a unicast router running at rate mR. And so, for copy multicasting, it is obvious that the speedup required increases in proportion to the number of copies made per cell, *i.e.*, its multicast fanout, m. For fanout multicasting, a single cell is sent to the center stage switches in case of a PPS (or memory buffers in the case of a single-buffered router), and no additional speedup is required to physically transmit the cell to the center stage switches (or write the cell to memory in case of an SB router). However, a higher speedup is required to ensure the existence of a layer in the PPS (or memory in case of an SB router) that satisfies the output constraints of all outputs to which the multicast cell is destined. Since there are m output link constraints, the speedup required is also proportional to m.[4]

Note that the two techniques are fundamentally orthogonal. Copy multicast does not use the fact that multiple output constraints could be satisfied simultaneously by transmitting a multicast cell to a common central stage OQ switch in the PPS (or a common memory in a single-buffered router). On the contrary, fanout multicast does not utilize the potential link/memory bandwidth available to make copies of multicast packets when they arrive.

J.3 A Hybrid Strategy for Multicasting

In any router, it is possible to implement either of the above methods. We will shortly show that the optimal strategy for multicast is to take advantage of both of the above methods. So we will introduce a technique called "hybrid multicasting".

Hybrid multicasting bounds the number of copies[5] that can be made from a multicast cell. We define a variable q as the maximum number of copies that are made from a given multicast cell. Note that since the maximum fanout of any given multicast cell is m, at most $\lceil m/q \rceil$ outputs will receive a copied multicast cell.

[4]A more detailed discussion on these observations, and the exact speedup required for fanout and copy multicasting, is available in our paper [230].

[5]In an earlier paper we referred to this as "bounded copy" multicasting.

In order to analyze a hybrid multicast strategy, we will need to find a lower bound on the size of the available input link set as a function of q. Before we proceed, recall that we defined a time slot for an SB router as follows:

ℵ**Definition J.3. Time slot:** This refers to the time taken to transmit or receive a fixed-length cell at a link rate of R.

Lemma J.1. *For a PPS that uses the hybrid multicast strategy,*

$$\boxed{|AIL(i,n)| \geqslant k - (\lceil k/S \rceil - 1)q, \forall i, n \geqslant 0. \qquad (J.1)}$$

Proof. Consider demultiplexor i. The only layers that i cannot send a cell to are those which were used in the last $\lceil k/S \rceil - 1$ time slots.[6] (The layer that was used $\lceil k/S \rceil$ time slots ago is now free to be used again.) $|AIL(i,n)|$ is minimized when a cell arrives to the external input port in each of the previous $\lceil k/S \rceil - 1$ time slots. Since a maximum of q links are used in every time slot, $|AIL(i,n)| \geqslant k - (\lceil k/S \rceil - 1)q, \forall i, n \geqslant 0$. ❐

J.4 A Load Balancing Algorithm for Multicasting

We are now ready to prove the main theorem, which first appeared in our paper [230].

Theorem J.1. *(Sufficiency) A PPS, which has a maximum fanout of m, can mimic an FCFS-OQ switch with a speedup of $S \geqslant 2\sqrt{m} + 1$.*

Proof. Consider a cell C that arrives at demultiplexor i at time slot n and is destined for output ports $<P_j>$, where $j \in \{1, 2, \ldots, m\}$. A maximum of q copies are created from this cell. We will denote each copy by C_y where $y \in \{1, 2, \ldots, q\}$. Each copy $<C_y>$ is destined to a maximum of $<P_j>$ distinct output ports, where $j \in \{1, 2, \ldots, \lceil m/q \rceil\}$. For the ILC and OLC to be met, it suffices to show that there will always exist a layer l such that the layer l meets all the following constraints for each copy C_y, *i.e.*,

[6]Recall that in a PPS the time slot refers to the normalized time for a cell to arrive to the demultiplexor at rate R.

$$l \in \{AIL(i,n) \cap AOL(P_1, DT(n,i,P_1)) \cap AOL(P_2, DT(n,i,P_2)) \ldots \\ AOL(P_{\lceil m/q \rceil}, DT(n,i,P_{\lceil m/q \rceil}))\},$$

which is satisfied when,

$$|AIL(i,n)| + |AOL(P_1, DT(n,i,P_1))| + |AOL(P_2, DT(n,i,P_2)) + \ldots \\ |AOL(P_{\lceil m/q \rceil}, DT(n,i,P_{\lceil m/q \rceil}))| > (\lceil m/q \rceil)k.$$

Since there are q copies, it is as if the AIL sees traffic at q times the line rate. Since each cell caters to $\lceil m/q \rceil$ AOL sets, the above equation is true if —

$$k - (\lceil k/S \rceil - 1)q + (\lceil m/q \rceil)(k - (\lceil k/S \rceil - 1)) > (\lceil m/q \rceil)k.$$

This will be satisfied if,

$$k - (\lceil k/S \rceil - 1)q - (\lceil m/q \rceil)(\lceil k/S \rceil - 1) > 0.$$

i.e., if,

$$(\lceil k/S \rceil - 1)(q + \lceil m/q \rceil) < k.$$

Note that the above analysis applies to each copy C_y that is made in parallel. Thus each copy C_y of the multicast packet has the same input link constraint, and by definition the same AIL. In the case that two or more distinct copies C_y, where $y \in \{1, 2, \ldots, q\}$, choose the same layer l, the copies are merged and a single cell destined to the distinct outputs of each of the copies C_y is sent. The speedup is minimized when $q + \lceil m/q \rceil$ is minimized. But $q + \lceil m/q \rceil < q + m/q + 1$ and so the minimum value is obtained when $q = \sqrt{m}$; *i.e.*, $S \geqslant 2\sqrt{m} + 1$. ❐

Theorem J.2. *(Sufficiency) A PPS that has a maximum fanout of m can mimic a PIFO-OQ switch with a speedup of $S \geqslant 2\sqrt{2m} + 2$.*

Proof. The proof is almost identical to the one above, but uses a PIFO-based analysis to compute the relevant AIL and AOL sets, similar to that described in Section 6.6.❐

The following theorem represents the worst-case speedup required to support multicasting in a PPS. It is a consequence of the fact that the maximum multicast fanout $m \leqslant N$.[7]

Theorem J.3. *(Sufficiency, by Reference) A PPS can emulate a multicast FCFS-OQ router with a speedup of $S \geqslant 2\sqrt{N} + 1$ and a multicast PIFO-OQ router with a speedup of $S \geqslant 2\sqrt{2N} + 2$.*

Theorem J.4. *(Sufficiency) A single-buffered router can emulate a multicast FCFS-OQ router with a speedup of $S \equiv \Theta(\sqrt{N})$ and a multicast PIFO-OQ router with a speedup of $S \equiv \Theta(\sqrt{2N})$.*

Proof. This is a direct consequence of Theorem J.3. In a single-buffered router (*e.g.*, the PSM, DSM, and PDSM), a packet is written to a memory that meets the AOL constraints of $<P_j>$ outputs, where, $j \in \{1, 2, \ldots, \lceil m/q \rceil\}$. Each copy C_y is automatically available to all the $<P_j>$ distinct output ports, when the copy C_y needs to be read by output j. There is no need for $\lceil m/q \rceil$ physical copies to be created in a center stage layer (as would need to be done by the OQ routers in the PPS if they maintained separate queues for each output), once a memory is chosen, since all outputs are assumed to have access to all memories. So the multicast speedup requirements are exactly the same as for the PPS router. Of course, N can be replaced by the maximum multicast fanout $m < N$ in the statement of the theorem. ❐

[7]Of course, in the above theorem, the variable N in the formulae for the speedup can be replaced by the maximum fanout of the multicast packets, m, if $m < N$. This is true in some routers.

J.5 A Caching Algorithm for Multicasting

In Chapter 7, we described a caching hierarchy to buffer and manage packets for high-speed routers. The caching technique can be trivially applied to copy multicasting, and this leads to the following theorem:

Theorem J.5. *(Sufficiency) For MDQF to guarantee that a requested byte is in the head cache (for a copy multicast router with maximum fanout m), the number of bytes that are sufficient in the head cache is*

$$\boxed{Qw = Q(m+1)\frac{b}{2}(3 + \ln Q). \qquad \text{(J.2)}}$$

Proof. This a direct consequence of Theorem 7.3. In a unicast router, there is one write and one read for every cell, and the memory access time is split equally between the writes and reads, leading to a memory block size of b bytes. In a copy multicast router, there are m writes for every one read, and so the block size b is scaled as shown above. ❒

We now consider fanout multicasting. Unfortunately, we do not currently know of any simple technique to support fanout multicast using caching. The main problem is that there is no way to associate a multicast packet with a specific queue, because fanout multicast packets are destined to multiple queues, all of which need access to it.

Another requirement in some routers is that they support in-order delivery of unicast and multicast traffic.[8] In such a case, the number of multicast queues that must be maintained in the buffer is $\Theta(2^m)$, where $m \leqslant q$ is the maximum multicast fanout [231], and q is the number of unicast queues. The cache size needed to support this feature is $\Theta(f * 2^f)$, and can be very large.

[8]Thankfully, this is not a common and mandatory requirement on most routers.

J.6 Conclusions

While we cannot predict the usage and deployment of multicast, it is likely that Internet routers will be called upon to switch multicast packets passing over very-high-speed lines with a guaranteed quality of service. A conclusion that can be drawn from the results presented here is that the speedup required to support multicast traffic grows with the size of the allowable multicast fanouts. With small fanouts at each switch, moderate speedup (or cache size) suffices and delay guarantees are theoretically possible. For large fanouts, the speedup (or cache size) may become impracticably large. As a practical compromise, we have implemented multicast buffering solutions [231], using techniques similar to frame scheduling (see Section 3.8). While we have made these compromises to enable high-speed multicasting, supporting in-order multicast with deterministic performance guarantees remains an interesting open problem.