Scheduling of Multicast Traffic in High-Capacity Packet Switches

Aleksandra Smiljanić
AT&T Laboratories, Middletown, NJ 07748, USA
phone: (732) 420-9052, fax: (732) 368-9477
email: aleks@research.att.com

Abstract— **As the traffic on the Internet grows, better quality of service should be provided to users. We have proposed earlier the weighted sequential greedy scheduling (WSGS) protocol that provides fast bandwidth reservations in terabit packet switches with input buffers. In switches with input buffers, a port that sources a multicast session might easily get congested as it becomes more popular. In this paper, we extend WSGS to support varying popularity of content on the Internet. Destination ports forward copies of multicast packets to other destination ports in a specified order. In this way, the multicast traffic load is evenly distributed over switch ports. The performance trade-off between capacity that can be reserved and guaranteed packet delay will be discussed.**

I. Introduction

As the Internet grows, high-capacity switches are needed. Also, the network should be more efficiently utilized, and better quality of service should be provided to users. For these reasons, explicit routing with bandwidth reservations and delay guarantees has been supported with frameworks such as RSVP and MPLS. Since applications on the Internet have a wide range of bandwidth requirements and holding times, high-capacity packet switches should be designed to support agile bandwidth reservations with fine granularity. Packet switches with input buffers can potentially provide high capacity because they require minimal buffering speed and switching fabric complexity [7]. But, they require a more complex controller than switches with output buffers where packets in different output buffers are scheduled independently [1], [7], [8]. We have proposed the weighted sequential greedy scheduling (WSGS) and an associated admission control protocol for switches with input buffers [9]. The implementation of this WSGS scales well by using a pipelining technique. The admission controller is agile since it performs a simple function. These features of the WSGS provide motivation to extend it to support multicast traffic as well.

The appeal of the Internet lies in the variety of services and content that it provides. A significant amount of traffic on the Internet is multicast in nature, i.e. transmitted from one source to multiple destinations (we will call them a multicast group). Switching of popular content through the Internet is troublesome. Today, the source usually sends a copied multicast packet separately to all destinations. In this case, the source and links close to it might become overloaded. Alternatively, multicast packets could be sent along precalculated multicast trees. Here, a packet is copied at branch nodes of the tree, so the transmission load is distributed over those nodes, and links closer to the source carry less traffic. The signalling and processing required to calculate these multicast trees is burdensome in wide area networks with a large number of nodes and edges [2], [3]. Assuming that the Internet growth has an upper bound, high-capacity switches would significantly reduce the number of nodes and edges in the network, and so more readily provide quality of service in wide area network. However, the processing bottleneck is moved to the single switch design.

It has been recognized that large switches with input buffers do not well support multicasting of popular content with large fan-outs (numbers of destinations). For example, it was shown in [5] that a three-stage Clos switch requires a speed-up equal to the maximum fan-out to ensure strict non-blocking. We have shown, [10], that the non-blocking condition in a cell-based switch with input buffers, and a three-stage Clos circuit switch are equivalent. So, a switch with a moderate speed-up would not carry popular multicast sessions properly. In addition, users attached to the port that multicasts popular content would be clogged. For this reason, we propose that copies of multicast packets should be forwarded through the switch by destination ports after they receive those packets. Packets are scheduled according to WSGS. In this way, simplicity and scalability of the WSGS and admission control protocols are maintained, and at the same time, the transmission load is balanced over different ports. We will discuss trade-offs between the capacity and delay required for packet forwarding, and show that a large portion of the port capacity may be reserved regardless of the traffic pattern. The proposed protocol flexibly supports changing popularity of different content on Internet. Implicitly, it allows a greater variety of content to be accessed by users, which adds value to the network.

II. Scalable Scheduling Protocol and Agile Admission Control Protocol in the case of Unicast Traffic

We proposed earlier a practical way to schedule unicast traffic in high-capacity switches [9]. Our approach is sequential greedy scheduling based on credits. We will

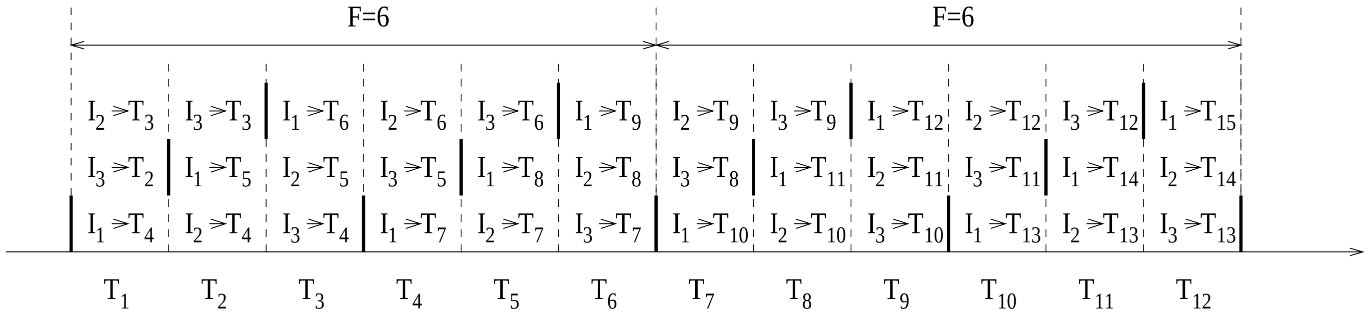

Fig. 1. Pipelining of the sequential scheduling protocol in a 3×3 switch.

present this approach for the sake of completeness. Inputs choose outputs one after another in a pipeline fashion. Namely, a schedule for one time slot is calculated in multiple earlier time slots, and multiple schedules are calculated in each time slot. Here, a schedule is the set of input-output pairs to be connected in a time slot, so that inputs in question transmit packets to outputs to which they are connected. Figure 1 shows the time diagram for pipelining where in each time slot, only one input selects an output for a particular time slot in the future. If $\mathrm{I}_i \rightarrow \mathrm{T}_k$ is assigned to some time slot T_j, it means that input I_i reserves an output for time slot T_k, and this reservation is made during time slot T_j. Bold vertical lines enclose the calculation of one schedule, which lasts N time slots in the given example. Here N denotes the number of input and output ports. In the more general case, in any time slot multiple inputs might select outputs for some future time slot, or it might take multiple time slots for an input to select an output for a future time slot. Time is further divided into frames comprising fixed number of time slots, F (as shown in Figure 1). In the specified time slot of a frame, counters of some input are set to the negotiated values. In the example shown, input i sets its counters c_{ij} to negotiated values a_{ij}, $c_{ij} = a_{ij}$, $1 \leq j \leq N$, in time slots $k \cdot F - N + i - 1$, $k \geq 1$. Only queues with positive counters would compete for service, and whenever a queue is served its counter is decremented by 1.

After inputs schedule packets from queues with positive counters, they might schedule packets from the remaining queues in the same pipelined fashion, as was described in [9]. In this way, the best effort traffic can be accommodated if there is some bandwidth left after the higher priority traffic is served. Packets are stored into different queues according to their destinations, so that the information about any queue status (empty or non-empty) and its heading packet is readily obtained. Such input buffer organization is often referred to as a buffer with virtual output queueing (VOQ) [1].

The pipelined sequential greedy scheduling algorithm is easy to implement, and it scales well with increasing number of ports and decreasing packet transmission time. An advantage of the proposed protocol is that it requires communication only among adjacent input modules, and, consequently, the simple scheduler implementation as shown in Figure 2. Also, by using pipelining the requirements on the speed of electronics are relaxed. In addition, it implies an extremely simple admission control protocol that provides agile bandwidth reservations. When bandwidth b_{ij} is requested by input-output pair (i, j), then $a_{ij} = \lceil b_{ij} \cdot F/B \rceil$ time slots per frame, i.e. credits, should be assigned to it. We have shown earlier ([9]) that the bandwidth can be allocated to input-output pair (i, j) if the following condition holds:

$$\sum_k a_{ik} + \sum_k a_{kj} \leq F + 1. \quad (1)$$

Consequently, the bandwidth can be allocated in a switch if for all $1 \leq i \leq N$ the following conditions hold:

$$T_i = \sum_k a_{ik} \leq \frac{F+1}{2},$$
$$R_i = \sum_k a_{ki} \leq \frac{F+1}{2}. \quad (2)$$

Here T_i is the number of credits assigned to input i for transmission, and R_i is the number of credits assigned to output i for reception. Half of the time slots per frame can be allocated to any input or output, meaning that 50% of the port capacity can be reserved for any unicast traffic pattern.

Simple admission conditions (2) allow fast bandwidth reservations in a switch with a large number of ports. Also, neither the scheduling protocol nor the admission control protocol depend on the frame length F. Since the frame length determines the granularity of bandwidth reservations $G = B/F$, the proposed protocols provide bandwidth reservations with fine granularity.

III. Performance Analysis of Flexible Multicasting in High-Capacity Switches

If the multicast packet could be scheduled for transmission to multiple destinations in the same time slot, the available bandwidth would depend on the multicast traffic pattern. Consequently, the admission control protocol would

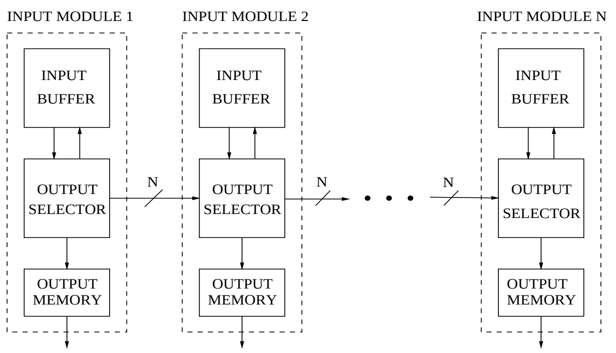

Fig. 2. Central controller implementing sequential greedy scheduling protocol.

be intricate, because it must consider the multicast traffic pattern. Also, some high-capacity switching fabrics do not allow the transmission of a packet to multiple destinations at the same time [4]. Alternatively, multicast packets might be independently scheduled for different outputs in the multicast group according to the described greedy algorithm. This algorithm is scalable, and implies the simple admission control protocol. However, if an input sends a multicast packet serially to all designated outputs, its bandwidth will be wasted in multiple transmissions of the same packet. Let's denote the number of time slots per frame assigned to multicast session (i,s) sourced by input i by a_{is}^m, the set of outputs in this multicast group by $\mathcal{M}_{is}$, and the number of outputs in set $\mathcal{M}_{is}$ by $|\mathcal{M}_{is}|$. Note that for a unicast session $|\mathcal{M}_{is}| = 1$. It follows from equation (2) that credits can be assigned to some input-output pair (i,j) if:

$$\sum_s a_{is}^m |\mathcal{M}_{is}| + \sum_{k,s,j \in \mathcal{M}_{ks}} a_{ks}^m \leq F+1. \tag{3}$$

In the worst case, input i sends packets to all N outputs, $|\mathcal{M}_{is}| = N$, where N is the number of ports, and from (3), the transmitting port is underutilized:

$$T_i = \sum_s a_{is}^m \leq \frac{F+1}{N}. \tag{4}$$

One Nth of the time slots in a frame can be allocated to input i, meaning that only $1/N$ of the transmitting port capacity can be utilized. Generally, utilization of the port capacity becomes low when a significant amount of multicast traffic that it transmits has a large fan-out. The performance degradation is more severe in high-capacity switches with a large number of ports, N.

Let us observe that once any port from the multicast group receives a multicast packet, it may as well forward it to $P \geq 1$ ports of that multicast group which have not received the packet. Here, each port comprises one input and one output. In this way, the transmission burden would be balanced over all ports in the multicast group. We have seen that for $P = N$, a multicast packet can be transmitted to all outputs within one frame, but a large number of credits might have to be allocated for this multicast session and the input port would get clogged. On the other hand, if each port forwards the packet to only one port, i.e. $P = 1$, then each port uses a small additional capacity for forwarding, but the multicast packet might experience a delay of up to N frames. Namely, in the worst case, a packet would be forwarded only once per frame. This delay would become excessive in high-capacity switches with a large number of ports N, and large frame lengths F. Apparently, there is a trade-off between utilized capacity and packet delay, that depends on the chosen parameter P.

We will analyze the switch capacity that can be guaranteed to the ports in terms of the parameter P. The bandwidth demand and packet forwarding order determine the credit allocation. It follows from equation (2) that credits can be assigned to some input-output pair (i,j) if it holds that:

$$T_i + E_i + R_j \leq F+1, \tag{5}$$

where

$$\begin{aligned} T_i &= \sum_k a_{ik}, \\ E_i &\leq P \cdot R_i = P \cdot \sum_k a_{ki}, \\ R_j &= \sum_k a_{kj}. \end{aligned} \tag{6}$$

Here, T_i is the total number of time slots per frame reserved for packets that are transmitted by input i, E_i is the number of time slots per frame reserved for input i to forward its multicast packets, and R_i is the number of time slots per frame reserved for packets bound to output i. Conditions (5,6) imply that credits can be assigned to input-output pair (i,j) if:

$$T_i + (P \cdot R_i) + R_j \leq F+1. \tag{7}$$

It further follows that the bandwidth allocation is possible if for all ports $i,\ 1 \leq i \leq N$, it holds that:

$$\begin{aligned} T_i &\leq F_t, \\ R_i &\leq F_r, \\ F_t + (P+1) \cdot F_r &= F+1. \end{aligned} \tag{8}$$

The total switching capacity that can be reserved equals:

$$C = N \min \left(F_t \cdot E[|\mathcal{M}|], F_r\right),$$

where $E[|\mathcal{M}|]$ is the average packet fan-out. Parameters $F_t,\ F_r$ are chosen so that the switch capacity is maximized for arbitrary traffic pattern:

$$C = \max_{F_t, F_r} \min_{E[|\mathcal{M}|]} \left(F_t \cdot E[|\mathcal{M}|], F_r\right) = \frac{F+1}{2+P}. \tag{9}$$

And, the bandwidth allocation is possible if for all ports, $1 \leq i \leq N$:

$$T_i \leq \frac{F+1}{2+P},$$
$$R_i \leq \frac{F+1}{2+P}. \quad (10)$$

So, the portion of the port capacity that can be reserved is $1/(2+P)$ in this case.

Next, we will calculate the packet delay in terms of the parameter P. Let us assume that a multicast packet of session (i,s) is forwarded to all outputs within S frames. In the first frame, the port that receives the packet from an input, forwards it to P ports. In the next frame, each of these ports forwards the packet to P^2 other multicast ports. In the last frame the packet is sent to at most P^{S-1} remaining ports. It holds that:

$$\begin{aligned} |\mathcal{M}_{is}| &> 1+P+\ldots+P^{S-2} = \frac{P^{S-1}-1}{P-1} \Rightarrow \\ S &< \log_P\left((P-1)\cdot|\mathcal{M}_{is}|+1\right)+1. \end{aligned} \quad (11)$$

For $P=2$ and $N=1024$, the maximum packet delay equals $S=10$ frames. There is an obvious trade-off between granularity $G=B/F$ and packet delay $D=S\cdot F\cdot T$, where T is the packet transmission time [10]. If we further assume $B=10$Gbps, $T=50$ns and $F=10^4$, the granularity of bandwidth reservations is $G=1$Mbps, and the packet delay is $D=5$ms. Since packets would pass through a small number of high-capacity switches, this packet delay could be tolerated even by delay-sensitive applications (voice and video conferencing). Finer granularity can be readily provided to applications which are less sensitive to the delay, as we will elaborate later. For $P=2$ the portion of the port capacity that can be reserved is 25%, regardless of the traffic pattern. The popularity of different content can vary arbitrarily in magnitude and over time. In [6], an unfortunate multicast traffic pattern was found for which the capacity utilized by greedy scheduling algorithms drops below 40% for large switches. So, the admission controller based on scalable scheduling protocols must consider multicast traffic patterns in order to utilize a larger portion of the switch capacity for more fortunate traffic patterns. However, our proposed protocol implies a very simple admission control that only checks port loads in order to allow new bandwidth reservations and still utilizes a significant portion of the switching capacity. The proposed admission control further simplifies provisioning, because the network planners should only ensure that the aggregate transmission and reception capacities of users attached to some port do not exceed specified values, without having to estimate exact traffic patterns.

IV. Scalable Scheduling Protocol and Agile Admission Control Protocol in the Case of Multicast Traffic

For fixed forwarding order, each port has to store ports to which multicast packets should be forwarded. It is not immediately clear how to determine the forwarding tree for a multicast session when $P \geq 2$. We propose the following simple algorithm for adding and removing a port to the tree. Each port of a tree should store the parent (previous) port and children (next) ports. Each port should also store its branch fan-outs, where the branch fan-out is the number of ports that could be reached through that branch. A request for adding a port to the multicast group is sent to the tree root. It, then, travels through the tree always taking the branch with the smallest fan-out. The fan-out of every branch that this request passes is increased by one, and the new port is added as a leaf to the tree (the port without children). Similarly, when a port wants to leave a tree it sends a request to the tree root. This request now travels through branches with the largest fan-outs until it gets to a leaf, and the fan-outs of these branches are decremented by one. The port to leave sends, along with the request, information about its parent and children ports, as well as about its branch fan-outs, so that the chosen leaf would store these parameters. Then, this leaf port informs its parent to stop forwarding packets to it, and the parent of the port leaving to start forwarding packets to it. We believe that in this way, minimal memory and processing per port are required for tree calculation and updates.

In the previous section we showed that the credits can be allocated to a new multicast session (i,n) if the following $|\mathcal{M}_{in}|+1$ inequalities are fulfilled:

$$a_{in}^m + \sum_k a_{ik} \leq \frac{F+1}{2+P}, \quad (12)$$

$$a_{in}^m + \sum_k a_{kj} \leq \frac{F+1}{2+P}, \quad (13)$$

for $j \in \mathcal{M}_{in}$. If bandwidth is requested for a new multicast session, admission conditions (12,13) are checked, and bandwidth is reserved accordingly. In the more general case, only a subset of multicast outputs have enough spare capacity, and they are admitted. Assume that the bandwidth is reserved for multicast session (i,n), and that the admitted multicast group of outputs is $\mathcal{M}_{in}^a$. The tree is constructed out of the admitted multicast group according to the described algorithm. Assume that source i transmits packets to port $p(i)$, and port j forwards packets to ports $p_k(j),\ 1 \leq k \leq P$. If a new multicast session (i,n) is admitted, credits are updated like:

$$a_{ip(i)} \leftarrow a_{ip(i)} + a_{in}^m, \quad (14)$$

$$a_{jp_k(j)} \leftarrow a_{jp_k(j)} + a_{in}^m, \quad (15)$$

for $j, p_k(j) \in \mathcal{M}^a_{in}, 1 \leq k \leq P$. Similarly, when the multicast session is released, the following updates are made:

$$a_{ip(i)} \leftarrow a_{ip(i)} - a^m_{in}, \tag{16}$$

$$a_{jp(j)} \leftarrow a_{jp_k(j)} - a^m_{in}, \tag{17}$$

for $j, p_k(j) \in \mathcal{M}^a_{in}, 1 \leq k \leq P$. It is also a realistic scenario that one or more ports request to join an already existing multicast session. They will be admitted if (13) is fulfilled and added to the tree as described. Credit allocation is done according to (15). Similarly, it may happen that some ports want to leave the multicast session. They are removed from the tree as described, and credit allocation is updated according to (17).

The admission of a multicast session can also be pipelined. In addition, the multicast session may be released in a pipelined fashion. Such pipelined admission control might better utilize the available bandwidth. For example, the bandwidth for a multicast session is reserved in one frame according to (14,15), but packets are transmitted only to the first port of the forwarding sequence in the next frame. So, the bandwidth reserved for forwarding of these multicast packets to the rest of the ports is wasted because they have not arrived into the appropriate queues yet. But, since the input transmits packets to the first port in the multicast group within one frame, then the bandwidth for forwarding packets by this port should be reserved in the same frame (which is one frame after the bandwidth has been reserved for transmission from input), and so on. Similarly, when a multicast session has ended, the input will stop transmitting packets, but packets that were previously transmitted might still be forwarded by the switch ports. So, the bandwidth should be released according to (16,17) $|\mathcal{M}_{in}|$ frames after the termination of the multicast session. Alternatively, the bandwidth reserved for forwarding of multicast packets from the first port in a forwarding sequence, should be released one frame after the bandwidth reserved for transmission from the multicast input has been released, and so on. The pipelined admission control can be summarized as follows. Input i reserves the bandwidth for transmission to port $j \in \mathcal{P}_1 = \{p(i)\}$ by updating the assigned credits according to (14) in some frame t if conditions (12) and (13) for $j \in \mathcal{P}_1$ hold. Then, port $j \in \mathcal{P}_1$ reserves bandwidth for packet forwarding to ports $j \in \mathcal{P}_2 = \{p_1(j), ..., p_P(j)\}$ for which conditions (13) hold, by updating the assigned credits according to (15) in frame $t+1$. In general, ports $j \in \mathcal{P}_l$ reserve the bandwidth for packet forwarding to the associated ports $j \in \mathcal{P}_{l+1} = \{p_k(j) | j \in \mathcal{P}_l, 1 \leq k \leq P\}$ for which conditions (13) hold, by updating the assigned credits according to (15) in frame $t+l$. Similarly, if this multicast session ends in frame t, input i releases the bandwidth reserved for port $p(i)$ in frame t, and ports $j \in \mathcal{P}_l$ release bandwidth reserved for forwarding packets to their associated ports $j \in \mathcal{P}_{l+1}$ in frame $t+l$.

At the beginning of each frame, counters associated with input-output pairs are set to their negotiated numbers of credits, $c_{ij} = a_{ij}$, $1 \leq i, j \leq N$. Packets are scheduled according to the previously described pipelined sequential greedy algorithm in which queues with positive counters are served with priority.

V. Conclusion

We proposed flexible bandwidth reservations for the multicast traffic in a high-capacity switch. In a network with a large number of low capacity switches, bandwidth reservations are hindered by the required signalling and high complexity admission control algorithms. Our protocol takes advantage of today's high capacity switching fabrics. Proposed centralized scheduler can make fast decisions and provide agile resource allocation. Multicast packets are forwarded through the switch, so that their transmission is balanced over the switch ports. The utilized switch capacity somewhat drops due to this packet forwarding, but the switch transports contents whose popularities change arbitrarily in magnitude and over time.

Aknowledgement: I thank Nick Frigo, Sheri Woodward and Bob Doverspike from AT&T Labs for their comments.

References

[1] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Transactions on Computer Systems*, vol. 11, no. 4, November 1993, pp. 319-352.

[2] S. Chen, K. Nahrstedt, and Y. Shavitt, "A QoS-aware multicast routing protocol," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, December 2000, pp. 2580-2592.

[3] C. Chiang, M. Sarrafzadeh, and C. K. Wong, "Global routing based on Steiner Min-Max trees," *IEEE Transactions on Computer Aided Design*, vol. 9, no. 12, December 1990, December 1990, pp. 1318-1325.

[4] J. Gripp, P. Bernasconi, C. Chan, K. L. Sherman, and M. Zirngibl, "Demonstration of a 1Tb/s optical packet switch fabric (80*12.5GB/S), scalable to 128 Tb/s (6400*20Gb/s)," *Postdeadline Paper in ECOC 2000.*

[5] M. Listanti, and L. Veltri, "Non blocking multicast three-stage interconnection networks," *IEEE GLOBECOM*, Decemeber 1999, pp. 1401-1405.

[6] A. M. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "On the throughput of input-queued cell-based switches with multicast traffic," *Proceedings of IEEE INFOCOM*, 2001, pp. 1664-1672.

[7] N. McKeown *et al.*, "The tiny tera: a packet switch core," *IEEE Micro*, vol. 171, Jan.-Feb. 1997, pp. 26-33.

[8] A. Mekkittikul, and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," *IEEE INFOCOM*, March 1998, pp. 792-799.

[9] A. Smiljanić, "Flexible bandwidth allocation in terabit packet switches," *Proceedings of IEEE Conference on High Performance Switching and Routing*, (Best Paper Award), June 2000, pp. 233-241.

[10] A. Smiljanić, "Flexible bandwidth allocation in high-capacity packet switches," *to appear in IEEE/ACM Transactions on Networking*, April 2002.