A Distributed Algorithm for Multicast Scheduling: a Way to Improve Performances of Multicast Protocols

J. Ignacio Alvarez-Hamelin* and Pierre Fraigniaud†

Laboratoire de Recherche en Informatique – CNRS
Université Paris-Sud
91405 Orsay, France
e-mail: {ihameli,pierre}@lri.fr

Abstract

Most of the current protocols handling multicast groups, *e.g.*, CBT or PIM, are based on tree-networks connecting all the members of the group. Multicast schemes make use of the edges of the tree to forward messages from the source to the destinations. In this paper, we present an algorithm based on an algorithm by Slater, Cockayne, and Hedetniemi, which allows to optimize the scheduling of the calls along the edges of the tree in order to minimize the latency of the multicast. Our algorithm can be run on top of existing multicast protocols. Given a tree T, our algorithm computes, at every node x of the tree, the best way for x to call its children in T so that the global latency of the multicast is minimized. The algorithm can be tuned in order to take into account not only the structure of the multicast tree, but also the traffic sharing resources with the multicast group. We have shown that usual multicast protocols can take benefits from our strategy, and can get significant improvements of their latency. The significance of the improvement is related to the ratio of the switching time over the propagation time of the links.

Key words: multicast, multi-point application, self-similar traffic.

Résumé

La plupart des protocoles récents pour les communications multipoints, comme par exemple CBT ou PIM, utilisent des arbres qui interconnectent les membres du groupe. Ces arbres sont utilisés pour transmettre les messages, de la racine vers les membres du groupe. Dans cet article, nous présentons un algorithme distribué basé sur un algorithme de Slater, Cockayne et Hedetniemi, et qui permet d'optimiser l'ordonnancement des appels le long des liens de l'arbre afin de minimiser la latence de la communication multipoint. Notre algorithme peut être mis en œuvre au dessus de protocoles multipoint existants. Etant donné un arbre T, notre algorithme calcule, pour tous le nœud x de l'arbre, le meilleur ordre dans lequel x doit appeler ses fils afin de minimiser la latence globale. Notre algorithme tient compte non seulement de la structure de l'arbre, mais aussi du trafic le long des liens de l'arbre. Nous montrons que les protocoles multipoint actuels peuvent tirer bénéfice de notre stratégie, et améliorer ainsi d'une manière significatif leur latence. Cette amélioration est cependant relative au rapport entre le temps de traitement d'un paquet et le temps de propagation sur un liens.

*Also: Facultad de Ingeniería, Universidad de Buenos Aires, Paseo Colón 850, (1063) Buenos Aires, Argentina.
Additional support from the *Alpha* program.

†Supported by the program A.R.P. of the CNRS, and by the French RNRT program "ROM".

1 Introduction

Many protocols for Internet multicast have been proposed in the literature, and some are even currently implemented in existing systems. One of their common features is to build a tree T connecting the memberships of the multicast group sharing a same multi-point application. This tree is used to carry on messages broadcasted among the members of the group. That feature is particularly satisfied by CBT [1], PIM-SP [6], MIP [17], YAM [2], and QoSMIC [7]. When some member x needs to broadcast some message, this broadcast is either initiated directly by x, or x sends a request to some specific node r of the tree via a point-to-point connection, and this node r is then in charge of initiating the broadcast. For the purpose of our discussion, we assume *w.l.g.*, that the node performing the broadcast is the root r of the tree T, and that r initiates itself the broadcast along the edges of T. We are concerned by the time it takes to perform this broadcast.

To perform broadcast, the source r makes use of intermediate nodes to relay messages. Upon reception of a message, such an intermediate node forwards this message to the nodes of its subtree. Roughly speaking, the broadcast tree T of root r is composed of k subtrees $T_1, \ldots, T_k$ rooted at the k children r_i, $i = 1, \ldots, k$, of r in T. During the broadcast, r successively calls its k children. When a child r_i receives the message, it calls successively its children in T_i, and so on. The situation may be more tricky because of different abilities of the nodes of the tree, but, for the purpose of our discussion, we assume that all nodes supporting the multicast protocol perform as previously described. In such a framework, the order in which a node calls its children matters, and the time-performance is a function of this order.

Several solutions have been proposed to optimize the broadcast time of a tree. Slater, Cockayne, and Hedetniemi [21] have described a centralized polynomial algorithm to compute the broadcast time of a tree in the following context: the network is synchronous, and, at each round, a sequence of different calls between neighboring vertices are performed. The restriction is that a node can participate to at most one call at each round. This model yields a $\lceil \log_2 n \rceil$ lower bound of the number of rounds required to perform a broadcast in any n-node network since the number of nodes aware of the message can at most double at each round. Later, Koh and Tcha [11] have improved the algorithm of slater *et al.* by considering weighted trees. They gave an $O(n \log n)$-time centralized algorithm to compute the broadcast time of a weighted tree in the same context as Slater *et al.*. Other results are connected to broadcasting in trees, although in an even less practical setting. Labahn [13], and Harutunian [10] have independently shown that the maximum, taken over all nodes, of the minimum number of rounds required to perform a broadcast from a node of any given tree is at least $1.44 \ \log_2 n$, and that there are trees achieving this bound. (Labahn [12] has also studied the minimum number of calls required to broadcast in a tree.) Broadcasting in trees has also been studied in a slightly different model assuming that calls can be placed between non-neighboring nodes under the constraint that two simultaneous calls be edge-disjoint. We mostly refer to Farley [8], Cohen, Fraigniaud, König and Raspaud [4], Cohen [3], and Cohen, Fraigniaud, and Midjana [5] for results in this context.

As far as a practical use is concerned, all these results suffer from two major drawbacks: firstly the algorithms are centralized, and secondly the communication model assumes very simple hypotheses on the traffic, and on the nodes and links performances. The aim of this paper is to show that these two problems can be overcame, and that important improvements can be practically obtained by an adaptation of some of these algorithms to real networks.

We present a distributed scheduling algorithm called MSDA (for *Multicast-Scheduling Dis-*

tributed Algorithm) that allows to schedule the calls in order to optimize the total delay of the multicast protocol. The algorithm takes into account the topology of the tree, the nature of the traffic, the performances of the nodes, and the characteristics of the links. In particular, we considerer the case in which the network support different traffics, and we applied the theory of self-similar traffic [16]. The algorithm performs as well in a context in which the tree dynamically changes along the time.

This paper is structured as follows. In Section 2, we present our distributed algorithm in a very basic setting, and we present its main properties. In Section 3, we compute the performances of this algorithm, still assuming very simple hypotheses on the network, and on the traffic. In Section 4, we improve our model to make it conform with the reality of existing networks and traffic. Section 5 revisits our algorithm to adapt it to this model. It also presents the analysis of MSDA in the setting of Section 4. Finally, Section 6 presents some concluding remarks.

2 A distributed algorithm to broadcast in trees

To simplify the presentation of our algorithm, let us start with a very simple model, often call "constant model" in the literature [9]:

- All links have the same transmission cost (this cost will be assumed to be 1);
- The processing time is negligible in front of the transmission cost (and therefore assumed to be zero);
- Single port communications (*ie.*, a node cannot send more than one message at a time).

The multicast group is assumed to be composed of all the nodes of the tree, and, as said in Section 1, the source is assumed to be the root of the tree. Assuming the destination nodes to form a subset of all nodes of the tree does not significantly change the idea and the performances of our algorithm, though it will make it more difficult to present, just because of technical details.

2.1 MSDA in a basic setting

MSDA has two components: the Scheduling Algorithm (SA), and the Scheduled Multicast Algorithm (SMA). SA computes the scheduling of the calls for the purpose of optimal multicast time. SMA is the application of SA. It performs the calls according to the result of SA. SA is based on Slater, Cockayne and Hedetniemi algorithm [21].

Consider a tree $T = (V, E)$ where V is the set of nodes (or routers), and E is the set of communication links. Since all nodes of the tree are able to run a multicast protocol, the dynamical evolution of the tree consists to add one node (which becomes a new leaf) or to remove a leaf of the tree. Let r be the root of the tree. The scheduling algorithm runs concurrently on all nodes of the tree. Each node v maintains a scalar variable t_v, and a vector variable σ_v according to the instructions given in Table 1.

Figure 1 shows an example of an execution of SA on a tree of 12 nodes. Assume first that the tree is given statically, that is let us concentrate on the procedure "compute" of SA. All leaves are marked 0. The single node f with a unique child is marked 1 since it takes a single round

Main. Every node v performs:

1. By sending a message "*change*", v advises its parent that it is joining the group ;
2. Set $t_v = 0$ and set $\sigma_v = Identity$;
3. Case:
   (a) v wants to leave the group: go to Step 4;
   (b) v receives a request from its parent: go to Step 5;
   (c) v receives a "*change*" from a child: go to Step 6;
   (d) v receives a "*recompute*" from its parent: go to Step 7;
4. v sends a message "*change*" to its parent, and STOP.
5. v sends the current value of t_v to its parent; Go back to Step 3;
6. Do
   (a) v has received an advise message "*change*" from a child w, a new node w, or a leaving node w; v sends to w a request for the new value of t_w;
   (b) Upon reception of the value of t_w ($t_w = -1$ is a non-to-be-used signal specifying that w has left the tree), v executes **compute**;
   (c) If t_v has changed, then v sends an advise message "*change*" to its parent; Go back to Step 3.
7. Do
   (a) v sends a message "*recompute*" to all its children (if any);
   (b) Upon reception of all the values t_w of its children, v executes **compute**;
   (c) v sends its value of t_v to its parent; Go back to Step 3.

Compute.

1. Let $t_1, t_2, ..., t_k$ be the current "t_{v_i}-values" of the k children $v_1, \ldots, v_k$ of v, ordered so that $t_i \geq t_{i+1}$, $1 \leq i < k$. Note that we do not necessarily have $t_i = t_{v_i}$.
2. Compute $t_v = \max_{1 \leq i \leq k} (i + t_i)$.
3. Set σ_v as the permutation on k symbols such that $t_{\sigma(i)} = t_{v_i}$, $1 \leq i \leq k$;

Table 1: Scheduling Algorithm (SA)

to broadcast in the subtree of root f. The parent c of f has two children. One of those, node e, is marked 0, and the other, node f, is marked 1. Since $t_f > t_e$, we set $t_1 = t_f = 1$, and $t_2 = t_e = 0$. Thus $t_c = \max_i(t_i + i) = \max\{t_f + 1, t_e + 2\} = 2$. This corresponds to the fact that it takes 2 rounds to broadcast in the subtree of root c. The permutation σ_c specifies that node c must first send to f, and then to e, in order to achieve a broadcast time of 2 rounds. Node d is marked 2 (it has two leaves). Since $t_c = t_d$, we set arbitrarily $t_1 = t_c$, and $t_2 = t_d$. We get $t_b = \max_i(t_i + i) = \max\{t_c + 1, t_d + 2\} = 4$, and the reader can check that 4 rounds are indeed required to broadcast in the subtree of root b. Finally, applying the same rules, we get $t_a = 5$, that is the broadcast time of the tree of Figure 1.

Assume now that node x joins the group. It makes $t_e = 1$, $t_c = 3$, and t_b unchanged. That is the broadcast time of the tree is unchanged. If y joins the group now, it makes $t_e = 2$, $t_c = 3$, and t_b unchanged. The broadcast time of the tree is again unchanged. Note however that the permutation σ_c has been modified to achieve this time. Indeed, c must call e before f so that the

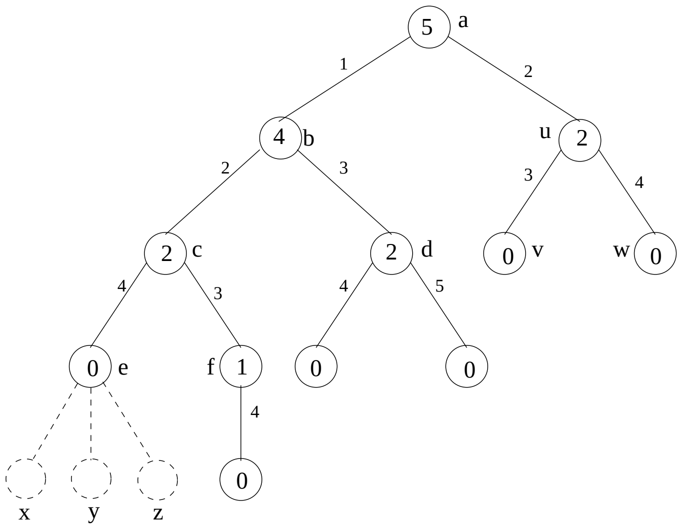

Figure 1: An example of the execution of SA on a tree of 12 nodes. The values of t_v are marked inside the nodes, and the rounds of the calls are indicated on the edges. Three additional nodes x, y, and z join successively the tree to form a 15-node tree.

SMA. Upon reception of a message, every node v performs:

> Send the message to all the children $v_1, \ldots, v_k$ according to the order defined by the permutation σ_v of the scheduling algorithm SA, that is v calls v_i at round $\sigma_v(i)$, or, in other words, at round i, v calls its child $v_{\sigma_v^{-1}(i)}$.

Table 2: Scheduled Multicast Algorithm (SMA).

broadcast time of the subtree of root c be 3. Finally, the arrival of a new node z increases the broadcast time of the tree by 1 round ($t_e = 3$, $t_c = 4$, $t_b = 5$, and $t_a = 6$). It would have been cleverer to add z elsewhere in the tree, as a child of a for instance, so that the broadcast time does not change. However, the way the nodes join the tree is not under our control since it depends of the group management protocol. This is not the role of our algorithm to optimize the placement of the nodes. Our algorithm is in charge of optimizing the communications once the placement has been set.

If the three nodes u, v, and w, leave now the group, the broadcast time of the tree will remain the same. However, if one of the three node x, y, or z leaves the tree, the broadcast time will decrease by one.

The scheduled multicast algorithm SMA runs in parallel with SA. It is given in Table 2.

2.2 Properties of MSDA

MSDA satisfies the following properties. First of all, the algorithm insures that the multicast scheme is performed with a minimal latency. This is a direct consequence of the fact that the centralized version by Slater *et al.* [21] satisfies this property. The distributed version SA does not modify this property. The local time-complexity of any update of t_v and σ_v is $O(k \log k)$ arithmetic operations for a node with k children, and the space complexity is $O(k)$. Indeed, the most complex operation consists to sort k integers. If a node is added or removed to or from the tree, only nodes along the path from the root to the new or old node may have to recompute their values of t_v and σ_v.

3 Efficiency of MSDA

Let us estimate the efficiency of MSDA in the basic setting of Section 2. Let us denote by $b(T,r)$ the minimum number of rounds required to broadcast a message from the root r of T to all the nodes of T. By construction, MSDA allows to broadcast optimally, that is in $b(T,r)$ rounds for any tree T. Without the use of MSDA, every node forwards a message to its children in an arbitrary order. Therefore the performance of MSDA is estimated by the ratio

$$\gamma = \frac{\bar{b}(T,r) - b(T,r)}{b(T,r)} \tag{1}$$

where $\bar{b}(T,r)$ is the average broadcast time from r in the tree T. Let S be the set of all the possible scheduling of the calls for broadcasting in T from r. We have $\bar{b}(T,r) = \frac{1}{|S|}\sum_S b_S(T,r)$ where $b_S(T,r)$ is the number of rounds required when broadcasting using the scheduling S. Note that we could have estimated the performance of MSDA by using the ratio $\frac{b_{max}(T,r)-b(T,r)}{b(T,r)}$ where $b_{max}(T,r) = \max_S b_S(T,r)$, but this would be unfair since the worst case could result from a very specific situation that never occurs in practice.

3.1 Arbitrary tree-structures

The following property allows to compare the parameters $b(T,r)$, $\bar{b}(T,r)$, and $b_{max}(T,r)$.

Property 1 *For any tree T rooted at r, $b(T,r) \leq \bar{b}(T,r) \leq b_{max}(T,r) \leq b(T,r)^2$, and there are rooted trees for which $b(T,r) = b_{max}(T,r)$, and rooted trees for which $\bar{b}(T,r) = \Theta(b(T,r)^2)$.*

Proof. The inequalities $b(T,r) \leq \bar{b}(T,r) \leq b_{max}(T,r)$ are straightforward. Let us show that $b_{max}(T,r) \leq b(T,r)^2$. Let Δ and D be the maximum degree of T, and the depth of T rooted at r, respectively. On one hand, $b_{max}(T,r) \leq \Delta + (D-1)(\Delta-1) \leq D\Delta$. On the other hand, $b(T,r) \geq \Delta$ and $b(T,r) \geq D$. Therefore $b_{max}(T,r) \leq b(T,r)^2$. The path P_n of n nodes, and of extremity r, satisfies $b(P_n,r) = b_{max}(P_n,r) = n-1$ because there is a single way to broadcast in P_n. Note that the star S_n of n nodes and center r also satisfies $b(S_n,r) = b_{max}(S_n,r) = n-1$ although there are $(n-1)!$ broadcast schemes in this network. Finally, let us consider the d-dimensional Binomial trees B_d as defined by Vuillemin [22]. Recall that B_0 is reduced to a single vertex called the root, and that B_d, $d>0$, is obtained from two copies of B_{d-1} by adding an edge between their two roots, and selecting one of these two roots to be the root r of B_d. It is not difficult to check that B_d has $n = 2^d$ nodes, and that $b(B_d,r) = d = \log_2 n$. On the other hand, Let $x_0, x_1, \ldots, x_d$ be the path from r to the single node s at distance d from r: $x_0 = r$, and $x_d = s$. We have $\deg(x_i) = d-i+1$ for $i \geq 1$, and $\deg(x_0) = d$. Therefore, going from x_i to x_{i+1} requires $\frac{d-i}{2}$ rounds in average, and hence $\bar{b}(B_d,r) = \sum_i \frac{d-i}{2} = \Theta(d^2) = \Theta(b(B_d,r)^2)$. ■

Property 1 says that an arbitrary scheduling of the calls of a multicast protocol can yield a multicast scheme whose delay is, in average, the square of what could be expected[1]. In other words, there are n-node tree-structures for which $\gamma \gg 1$, and actually for which γ grows as fast as $\sqrt{n}$: take trees consisting of a path of length $\sqrt{n}$ where each node of the path is connected to $\sqrt{n}$ other nodes. $b(T,r) = \Theta(\sqrt{n})$ whereas $\bar{b}(T,r) = \Theta(n)$.

One can argue however that trees constructed by most of the multicast protocols on Internet do not produce such worst case structures. We will see in the next section that SA can produce a significant improvement anyway, even for "standard" tree-structure.

3.2 Multicast trees

The parameter γ of Equation 1 is large when there is an asymmetry between the branches of the tree, that is when the lengths of the different branches strongly differ. Indeed, the probability of choosing a bad schedule at each node of the tree is large in these situations. CBT, PIM and BGMP use a shortest path technique for constructing the tree. YAM computes multiple routes, and chooses one route according to some *QoS* parameters. Similarly, the QoSMIC protocol is essentially based on *QoS* parameters. All these multicast protocols build asymmetric trees because they are used for networks which have non symmetrical topologies in general, and because they all construct the trees without any symmetrical considerations. The largest contribution to $b(T,r)$ comes from a long branch with nodes of large degree. Let us consider a network with degree Δ and diameter D. According to real network topology, one may assume $D \gg \Delta$. Thereafter, we consider two cases that we claim to be representative of multicast trees constructed by known multicast protocols.

Case 1. A long branch with constant (small) degree. Consider the case of a tree T in which all the nodes but the last of the longest branch (of length $h \gg 1$) have the same fixed degree $\Delta \geq 3$ (see Figure 2). We have $b(T,r) = h + \Delta - 2$ because the penultimate farthest node receives the message at round $h-1$, and it has $\Delta - 1$ nodes to inform.

On the other hand, $\bar{b}(T,r) = h\frac{\Delta-1}{2} + \frac{\Delta}{2}$ because the probability of choosing the correct link is $\frac{1}{\Delta-1}$ at each of the $h-2$ intermediate nodes of the longest branch, and $\frac{1}{\Delta}$ at node 0. Plugging these values in Equation 1, and computing the limit of γ when h increases, we get

$$\lim_{h\to+\infty} \gamma = \frac{\Delta - 3}{2}. \tag{2}$$

Case 2. A long branch with nodes of random degree. Assume that each of the h first nodes of the longest branch $v_0, \dots, v_h$ has a degree equal to 2 plus a random variable uniformly distributed in $[0, \Delta - 2]$, where Δ is a given constant ≥ 2. Then, the expected value of $b(T,r)$, denoted by $\mathbf{E}(b(T,r))$, satisfies $\mathbf{E}(b(T,r)) = h + \mathbf{E}(X)$ where

$$X = \max_{i=1,\dots,\Delta-2} (\deg(v_{h-i}) - i - 1).$$

Therefore, we have

$$\mathbf{E}(b(T,r)) = h + \frac{\Delta}{2} + o(h).$$

[1]As pointed out by Pelc [19], if we do not restrict our study to trees, one could have $\bar{b}(G,r) = \Omega(2^{b(G,r)})$ by taking G as the complete graph K_n plus one additional node connected to a single node of K_n.

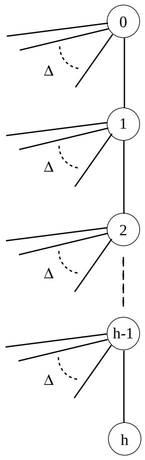

Figure 2: A branch of length h where all nodes (but the last) have degree Δ.

On the other hand

$$\bar{b}(T,r) = h + \sum_{i=0}^{h-1}(\frac{\mathbf{E}(\deg(v_i))}{2} - 1) + o(h),$$

that is

$$\bar{b}(T,r) = h(\frac{\Delta - 2}{4} + 1) + o(h).$$

Thus, we have

$$\lim_{h \to +\infty} \gamma = \frac{\Delta - 2}{4}. \tag{3}$$

Conclusion. In the simple setting of the model of Section 2, we have seen that there are cases in which the average behavior of a multicast algorithm can yields a delay equal to the square of the optimal. In more "reasonable" cases (that is cases appearing in practical situations), the improvement is less significant. However, Equations 2, and 3 show that an improvement of 33% on the latency can be obtained by using MSDA implemented above standard existing multicast protocols (taking $\Delta = 4$ in Equations 2 and 3 yields $\frac{\gamma}{\gamma+1} \geq \frac{1}{3}$).

The rest of the paper is dedicated to the implementation of MSDA for real networks, by taking into account both links and nodes performances, together with the traffic.

4 Network model

4.1 Generalities

As usual, the network is modeled by a symmetric directed graph $G = (V, E)$, where V represents the set of nodes, and E models the set of communication links. Each link of the network corresponds to two symmetric arcs e and e' in E, each arc corresponding to one direction of the link, that is we support full-duplex communications. Each vertex $v \in V$ actually represents a router of the network. A router is in turn modeled as follows (see Figure 3). A router is supposed to have one

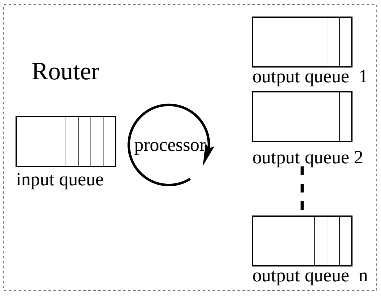

Figure 3: A router is composed of a single input queue, several output queues, and a processor in charge of the internal switching of the packets.

input queue, and n output queues, one for each output links (the degree of the corresponding node is n). These queues are FIFO. The distribution of the incoming messages over the output queues is performed by a dedicated processor.

There are several parameters associated to the links:

- $Bdw(e)$ is the bandwidth of link e in each direction (if a link has different bandwidths in its two directions, we can distinguish them if required);
- $Lat(e)$ is the propagation delay into the link e (it depends on both the length and technology of the link);

The two previous parameters are independent of the traffic inside the network. Also, there are several delay times associated to a router:

- $t_{input}(v)$ is the time that a packet waits in the input queue of router v (we consider that a packet is the minimal data structure that can be sent into the network);
- $t_{output}(v, e)$ is the time that a packet waits in the output queue corresponding to the output link e of v;
- $t_{switch}(v)$ is the processing time of a packet in v.

All these parameters, but $t_{switch}(v)$, depend on the traffic inside the network. The next section is dedicated to the traffic model.

4.2 Traffic and queue model

We assume the network traffic to be statistically self-similar [14]. This assumption is preferred to the Poisson model because observations of real traffics showed that the traffic does not satisfy the Poisson hypotheses [18, 20]. Indeed, a real traffic has a fractal behavior [14], and, at every time scale, from a few milliseconds to hours, one can observe similar-looking traffic bursts. In other words, we have the same statistical characteristics at different time scales, and this is typical of self-similar processes.

More precisely, we model the traffic $A(t)$ by using a *fractional Brownian traffic* $Z(t)$ [16]. $A(t)$ represents the total amount of bits offered to an input queue during the time-interval $[0, t)$ (with

$A(0) = 0$). It satisfies $A(t) = \mu t + \sqrt{a\mu}\, Z(t)$ where μ is average of the traffic, and a is the variance of the traffic. $Z(t)$ is a stochastic process with $t \in]-\infty, +\infty[$ that is characterized by the so-called Hurst (or self-similarity) parameter H, $H \in [\frac{1}{2}, 1)$.

Altogether, the traffic can be characterized by three parameters: μ, a, and H, where a and H somewhat measure the *quality* of traffic, and μ measures the *quantity* of traffic. Norros has shown in [16] that the maximum length Q_e of the queue of a link of bandwidth $Bdw(e)$, that is the maximum number of packets in the queue, is related to the coefficient ϵ of the packet loss ratio at a queue (i.e., 1 minus the probability that the length of the queue exceeds its size) by:

$$Q_e = \left(\frac{H}{Bdw(e) - \mu}\right)^{\frac{H}{1-H}} (1-H)(-2a\mu \ln \epsilon)^{\frac{1}{2-2H}}. \tag{4}$$

We will see later how this relation can help us to estimate $t_{output}(v, e)$.

5 MSDA in a realistic setting

5.1 Adaptation of MSDA to the traffic behavior

According to the model defined in Section 4, we replace Step 2 of the procedure "compute" of SA described in Table 1 by

2'. Compute $t_v = \max_{1 \le i \le k} (i\, t_{switch}(v) + t'_i)$, where t'_i is defined by

$$t'_i = t_{output}(v, e) + Lat(e) + \frac{\ell}{Bdw(e)} + t_i$$

where t_i is the time of the ith child node, e is the link connecting v to its ith child, and ℓ is the size of the message. The children of v are supposed to be ordered so that $t'_i \ge t'_{i+1}$.

This modification is motivated by the fact that it takes a time $t_{switch}(v) + T_{transfert}(v, e)$ for a message of size ℓ to be transmitted from a node v to a neighbor v' connected to v by the link e, where

$$T_{transfert}(v, e) = t_{output}(v, e) + Lat(e) + \frac{\ell}{Bdw(e)} + t_{input}(v').$$

For VLSI design reasons, the input queue is usually very short, and the waiting time $t_{input}(v')$ is often negligible. Thus, the term $t_{input}(v')$ is not used in the modification of SA.

Remark. In general, we have $\frac{\ell}{Bdw(e)} \ll Lat(e)$. For instance, in a campus-network, the bandwidth of the links is high and the distance between routers is small. Moreover, there exist many systems for which $Lat(e) \ll t_{switch}(v)$. Therefore, one can often assume that $T_{transfert}(v, e) \simeq t_{output}(v, e)$. There are cases however for which it is necessary to consider all the timing parameters.

All parameters but $t_{output}(v, e)$ are known in advance, that is they depend on the network only. They can be measured once, and some can be set according to the specifications of the network. The parameter $t_{output}(v, e)$ depends on the traffic, and therefore it cannot be fixed in advance. Let us figure out how to get a good estimation of $t_{output}(v, e)$ according to the self-similarity of the traffic. From Equation 4, if we know the maximum length Q_e of the queue of link e, and if we know in addition the size S_e of that buffer, then

$$t_{output}(v,e) = \frac{\min\{Q_e, S_e\}}{Bdw(e)}. \tag{5}$$

The value of S_e is known from the specification of the network. The value of Q_e is computed thanks to Equation 4 assuming that μ, a, and H are known, and that ϵ is fixed. A last problem still needs to be investigated, namely the frequency of refreshment of the values t_v (this is the aim of Step 7 in SA). Frequent refreshments are forced by the fast evolution of the traffic over short period of time, and in particular by possible change of the variance a.

Evolution of the traffic

The parameter H is actually constant for self-similar traffic, or at least is constant over a long period of time. It can be computed from the traffic by standard techniques [16]. The average μ of the traffic is also constant, but the variance a is susceptible to change along the time. Neidhardt and Wang [15] proposed the use of *relevant time scales* for queuing systems with self-similar traffic:

$$\tau = \frac{S_e}{Bdw(e) - \mu} \, \frac{H}{1-H} \tag{6}$$

that is the time for which the length Q_e of the queue could exceed the size S_e of the buffer. It was shown in [15] that τ is the best period for recomputing a (and possibly recomputing μ). As a consequence, we also set τ as the interval of time at which we periodically run SA to update the values of t_v and σ_v at every node.

5.2 Performances of MSDA as a function of $t_{transfert}(v,e)/t_{switch}(v)$

The performances of MSDA strongly depend on the ratio $t_{transfert}(v,e)/t_{switch}(v)$. Indeed, if $t_{transfert}(v,e) \ll t_{switch}(v)$, then the order in which a node forwards a message to its children does not matter, whereas it matters if $t_{transfert}(v,e)$ is of the same order of magnitude as $t_{switch}(v)$.

Let us be more precise, and let us analyze the latency induced by the longest branch $v_0,\dots,v_h$ of a tree T. Let us perform the same analysis as we did in Section 3 in order to derive the relative performance ratio. For the sake of simplicity, assume that $t_{switch}(v_i) = t_{switch}(v_j)$ for all i and j, and $t_{transfert}(v_i, e_{i'}) = t_{transfert}(v_j, e_{j'})$ for all i, i', j and j'. We get a performance ratio of

$$\beta = \frac{(\bar{b}(T,r)\; t_{switch} + h\; t_{transfert}) - (b(T,r)\; t_{switch} + h\; t_{transfert})}{b(T,r)\; t_{switch} + h\; t_{transfert}}. \tag{7}$$

Let $\alpha = \frac{t_{transfert}}{t_{switch}}$. We get:

$$\beta = \frac{\gamma}{1 + \alpha\, \frac{h}{b(T,r)}} \tag{8}$$

where γ is defined by Equation 1. Therefore, since $b(T,r) \geq h$, we have:

Property 2 *Let γ be the performance ratio of the scheduling algorithm SA in the constant model. The performance ratio of SA in networks is at least $\gamma\,\frac{1}{1+\alpha}$ where $\alpha = \frac{t_{transfert}}{t_{switch}}$.*

As a consequence of this property, and from the study of Section 3.2, our algorithm can bring significant improvements for networks in which $t_{transfert}$ and t_{switch} have the same order of magnitude (*e.g.*, a campus network). Moreover, if $t_{transfert} \gg t_{switch}$, then the improvement of SA will be significant for networks satisfying $\gamma \gg 1$. There are such networks. Indeed, we have seen in Property 1 that there are networks for which $\bar{b}(T,r) \gg b(t,r)$. The improvement of SA is high for these networks, even if $t_{transfert} \gg t_{switch}$.

6 Conclusions

We have proposed a distributed algorithm for scheduling the calls in a multicast tree in order to minimize the latency of the multicast schemes. This algorithm can be implemented on top of existing multicast protocols. The local cost of our algorithm is $O(k \log k)$ for a node of degree k, which is reasonable with the current technology. Our algorithm takes into account all physical parameters with significant influence: switching time, bandwidth and latency of the links, length of the queues, *etc.* Moreover, our algorithm takes into account the traffic inside the network. This is achieved by modeling the traffic as a self-similar process.

We have shown that the performance ratio β of the multicast scheme resulting from our algorithm compared to an oblivious multicast scheme is a function of the ratio $\alpha = \frac{t_{transfert}}{t_{switch}}$. Namely, $\beta = \gamma \frac{1}{1+\alpha}$ where γ is the performance ratio of our algorithm in the constant model. Since there are tree-networks satisfying $\gamma \gg 1$, there is a tradeoff between γ and $\frac{1}{1+\alpha}$ (this latter ratio can indeed be very small for networks satisfying $t_{transfert} \gg t_{switch}$). For networks in which $t_{transfert}$ and t_{switch} differ of, say, no more that one or two orders of magnitude, our algorithm allows significant improvements since, again, γ can be large, even for reasonably shaped tree-topologies.

References

[1] T. Ballardie, P. Francis, and J. Crowcroft. Core based tree (CBT): an architecture for scalable inter-domain multicast routing. In *proceedings of SIGCOM*, pages 85–95. ACM press, 1993.

[2] K. Carlberg and J. Crowcroft. Building shared trees using a one-to-many joining mechanism. In *ACM SIGCOM Computer Communication Review*, pages 5–11, January 1997.

[3] J. Cohen. Broadcasting, multicasting and gossiping in trees under the all-port line model. In *10th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 164–171. ACM press, 1998.

[4] J. Cohen, P. Fraigniaud, J-C. Konig, and A. Raspaud. Broadcasting and multicasting in cut-through routed networks. In IEEE, editor, *11th Int. Parallel Processing Symposium (IPPS)*, 1997.

[5] J. Cohen, P. Fraigniaud, and M. Mitjana. Scheduling calls for multicasting in tree-networks. In *10th ACM-SIAM Symp. on Discrete Algorithms (SODA '99)*, pages 881–882, 1999.

[6] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, and L. Wei. Protocol independent multicast-sparse mode (PIM-SM) RFC 2362. Technical report, IETF, 1998.

[7] M. Faloutsos, A. Banerjea, and R. Pankaj. Qosmic: Quality of service sensitive multicast internet protocol. In *SIGCOM*, September 1998. Vancouver BC.

[8] A. Farley. Minimum-time line broadcast networks. *Networks*, 10:59–70, 1980.

[9] P. Fraigniaud and E. Lazard. Methods and Problems of Communication in Usual Networks. *Discrete Applied Mathematics*, 53:79–133, 1994.

[10] H. Harutyunyan. Doctoral Thesis (in Russian), Armenia.

[11] Koh and Tcha. Information dissemination in trees with non-uniform edge transmission times. *IEEE Transaction on Computers*, 40(10):1174–1177, 1991.

[12] R. Labahn. The telephone problem for trees. *Journal of Information Processing and Cybernetics*, 22(9):475–485, 1986.

[13] R. Labahn. Extremal broadcasting problems. *Discrete Applied Mathematics*, 23:139–155, 1989.

[14] W. E. Leland, M. S. Taqqu, W Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic. In *SIGCOM '93*. ACM, 1993. San Francisco, USA.

[15] A. L. Neidhardt and J. L. Wang. The concept of relevant time scales and its application to queuing analysis of self-similar traffic. Technical report, Bellcore, http://www.cs.wpi.edu/~sigmet98/, 1998.

[16] I. Norros. On the use of fractional brownian motion in the theory of connectionless networks. *IEEE Journal of Selected Areas in Communications*, 13(6):953–962, August 1995.

[17] M. Parsa and J.J Garcia-Luna-Aceves. A protocol for scalable loop-free multicast routing. *IEEE Journal on Selected Areas in Communications*, 15(3):316–331, April 1997.

[18] V. Paxon and S. Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transaction on Networking*, 3(3):226–244, June 1995.

[19] A. Pelc. Private communication.

[20] P. A. Porras and A. Valdes. Live traffic analysis of TCP/IP gateways. In *Networks and Distributed Systemd Security Symposium.* Internet Society, March 1998.

[21] P. Slater, E Cockayne, and S. Hedetniemi. Information dissemination in trees. *SIAM Journal on Computing*, 10(4):692–701, 1981.

[22] J. Vuillemin. A data structure for manipulating priority queues. *Comm. Ass. Comput. Mach.*, 21:309–315, 1978.