An Active Layered Multicast Adaptation Protocol

Lidia Yamamoto and Guy Leduc

University of Liège, Research Unit in Networking
Institut Montefiore, B28, B-4000 Liège, Belgium
yamamoto@run.montefiore.ulg.ac.be
leduc@montefiore.ulg.ac.be
WWW home page: http://www-run.montefiore.ulg.ac.be

Abstract. We describe an active application in the field of multicast congestion control for real-time traffic. Our Active Layered Multicast Adaptation Protocol is a layered multicast congestion control scheme built on top of an Active Network infrastructure. It benefits from router support in order to obtain information about resources available and to perform the adaptation tasks at the places where shortage of resources occur. It supports heterogeneous receivers through the combination of layered multicast transmission with selective filtering and pruning of layers within the active nodes. Market-based resource management ideas are applied to achieve a resource utilisation level that represents an equilibrium between the user goals and the node operator goals. Our simulation results show that the protocol is feasible and provides adequate reactions to short term and persistent congestion, while keeping the amount of state and processing in the active nodes limited.

1 Introduction

We are interested in the use of active networks (ANs) [4, 22] in the context of adaptive multicast protocols. Since no single solution for multicast seems able to satisfy all applications, it seems more natural to adopt an approach in which multicast protocols can be dynamically loaded according to the needs of the applications and users. Active networks are especially suited for this task.

An adaptive protocol must be able to make optimal use of the available resources, and to accommodate fluctuations in resource availability. Here again, active networking can play an important role, since it becomes possible to inject customised computations at optimal points in the network, to facilitate the adaptation process.

In this paper we describe an active application (AA) in the field of multicast congestion control. We consider a soft real-time, distributed multicast application composed of sources and receivers that exchange capsules containing mobile code which is executed in the active nodes along the path. Our Active Layered Multicast Adaptation (ALMA) protocol supports heterogeneous receivers, i.e.

Second International Working Conference on Active Networks (IWAN 2000)
H. Yasuda (Ed.): LNCS 1942, pp. 180-194. Tokyo, Japan, October 2000.

receivers that have different terminal characteristics or experience different network conditions, by making use of layered multicast [17]. In a layered multicast scheme, the source divides its data into a number of layers in a hierarchical way, such that the most important information is carried by the lowest layer (base layer), and the higher layers successively enhance this information.

The rest of the paper is organised as follows: Section 2 provides some background on research related to this work, Sect. 3 describes the ALMA AA, Sect. 4 shows some results obtained so far, and Sect. 5 concludes the paper.

2 Background

2.1 Resource management

One of the main fears towards active networks is the time dedicated to mobile code processing: this seems incompatible with the increasing bandwidth availability in the network backbone, which requires faster and therefore less intelligent routers. However, the access networks do not follow the growing speed of the backbones. There, heterogeneity is becoming even more prominent, including fixed and mobile dial-up access, ISDN, xDSL, cable networks, etc. A price measure seems to be a fairly good reference to reflect these differences in availability and cost of resources. Applying simplified economic principles of offer and demand, it is possible to make the price indicate the availability of a given resource. Note that the price is only an arbitrary reference that enables comparisons and trading between different types of resources, thus does not necessarily translate to real-world prices. For example, an application could trade CPU time for bandwidth consumed while crossing a high-bandwidth backbone (where CPU processing is expensive and bandwidth is cheap), while it could favour CPU consumption where bandwidth is scarce (thus expensive) and CPU time cheap.

This gives rise to a market-based model for resource allocation. In such a model, two opposite forces act to seek the global system equilibrium by optimising their own benefits: on one side we have the network elements, whose interest is to maximise resource usage (since that brings them revenues) while maintaining a good performance level (in order to keep the clients satisfied). On the other side, we have the users (active applications), who seek to obtain a better quality/price relation for the resources consumed, and to efficiently manage their own budgets avoiding waste.

Several algorithms inspired on operational research and economy theories have been proposed to control resource usage in networks [7]. These algorithms are able to converge towards a globally optimal resource allocation in a decentralised way. In [9, 14, 15] such theories are applied to the problem of end-to-end congestion control, i.e. where bandwidth is the main scarce resource. In [9] an optimisation framework is used to derive a class of optimal algorithms inside which TCP, after some modification, can be seen as a special case. In [14] a thorough stability and fairness analysis of some optimisation-based rate control

algorithms is presented, and it is shown that these algorithms implement proportionally fair pricing. In [15] a similar algorithm is proposed, and in a more recent work [1] it is adapted to the Internet environment, using a packet tagging scheme to communicate link price information to the end hosts. The results shown are promising since they are generic enough to be adapted to a wide variety of applications. However, their direct application to discrete layering multicast schemes such as the one we present in this paper, is not straightforward due to fairness issues, as pointed out in [20].

Closer to the AN perspective, in [23] an open resource allocation scheme based on market models is applied to the case of memory allocation for mobile code. In [11] an adaptive QoS scheme for MPEG client-server video applications is described. It is based on intelligent agents that reserve network bandwidth and local CPU cycles, and adjust the video stream appropriately. In [26] a market model to allocate QoS is applied to a conferencing tool targeted at casual meetings where sudden variations in bandwidth availability require an adaptive QoS control strategy.

A cost model for active networks is proposed in [18], which expresses the trade-off between different types of resources in a quantitative way. However, due to the recursive approach adopted, the model seems more appropriate for applications that make use of resource reservation, instead of highly adaptive ones. Furthermore, the use of such model with multicast needs to be clarified.

2.2 Multicast Congestion Control

Congestion control can be regarded as a special case of resource management which considers mainly bandwidth as a scarce resource. It has been mentioned several times as a potential area of application that can benefit from active networks. End-to-end congestion control is an extremely difficult problem, particularly on the Internet where it is aggravated by the absence of information about the network. In an active network, capsules can collect information about network conditions [5] and active filters can be installed to adapt between portions of the network subject to different conditions [13]. But to which extent such an additional information is helpful, or the extra burden it might bring, are still open questions.

Multicast sessions can be either single rate, if a single flow is generated for all the receivers, or multirate, when several flows are generated and the receivers get a subset of flows according to their preferences and constraints. If these flows are alternatives to each other, we have a simulcast scheme. If they complement each other in a hierarchical way, we have a layered scheme. Layered multicast is typically proposed for video, but it has a potential for any kind of application that can divide a single flow into subflows of different priorities, e.g. an animation stream, bulk data transfers, etc.

Multicast congestion control schemes can be divided into two classes: pure end-to-end schemes, and router-assisted schemes. In a pure end-to-end layered multicast scheme [17, 24], the source transmits a hierarchically encoded stream, and the receivers subscribe/unsubscribe to a number of layers according to the

observed network conditions (loss rate, etc.). The RLM protocol [17] is one of the pioneers and most well-known protocols in this category. It is based on the so called *join experiments* to probe for available bandwidth, and on the observed loss rate to abandon layers in case of congestion. RLC [24] is a more recent protocol that also addresses friendliness towards TCP flows on the Internet.

The existing end-to-end layered schemes achieve very good results considering the limitations of pure best-effort networks which make most of the Internet today. However, they have many drawbacks such as: slow and/or coarse-grain adaptation; unstable behaviour characterised by subscribe/unsubscribe oscillations; the need to allocate and manage several multicast groups; random packet drops that lead to poor quality due to hierarchical dependence among packets from different layers; probing for additional bandwidth has a potential to intensify congestion; difficulty to synchronise among receivers leading to underutilisation of bandwidth.

Recently, more attention has been devoted to router-assisted schemes [3], that can count on router support in order to solve the abovementioned problems. In [10] an extension of RLM is proposed, which makes use of two priority levels: the upper layer, which is less important than the others, is always assigned a low priority. This creates a "bumper layer" that absorbs most of the packet losses. Unnecessary join experiments are eliminated in this way, leading to a very stable behaviour even when the layers have a variable bit rate profile.

In [16] a router-assisted congestion control scheme for reliable multicast is sketched. The authors propose a very interesting filtering scheme based on a normalised inverse sequence number which is mathematically the smoothest scheme possible. They assume that the router computes the fair shares for all the sessions. Their signalling scheme is similar to ours, but an explicit rate is used instead of layers.

For a router-assisted scheme to be useful, it needs to be widely accepted and deployed, or at least deployed in critical points in the network, which anyway requires a long standardisation process and deployment time. Since most multicast sessions are naturally heterogeneous, it is difficult to agree on a single solution or set of solutions. That is where active networks can play a role, since we can design customised solutions and let them evolve through usage experience.

In [6] a unicast scheme for AN-based congestion control is presented, targeted to improve TCP performance over links presenting high bandwidth delay product. In [12] a single rate reliable multicast scheme with congestion control is presented in the form of an active service. Within this scheme, the source adapts its sending rate to the rate that can be supported by the "nominee", which is the weakest receiver in the session. In [2], a layered video multicast protocol is implemented over two AN platforms: ANTS and M0. Although the goal of that work was to compare the performance of the two AN platforms, it is important to notice that their signalling scheme to join layers is identical to ours.

A lot of AN example applications are related to information filtering or transcoding in the active nodes [13, 21]. The common idea to these examples is to show how more intelligent functionality such as selective packet discard-

ing and transcoding in the routers can help improving the reception quality of a multimedia stream. In all these examples, the source of congestion does not react in order to prevent overflow and discarding. Filtering in the intermediate active nodes alleviates downstream congestion, and is important for heterogeneous multicast, since it enables the adaptation of one original stream for several different groups of receivers. However, filtering alone is obviously not a solution to the congestion control problem, since it does not tackle the source of the congestion.

3 ALMA Protocol

Active Layered Multicast Adaptation (ALMA) is a multicast congestion control protocol implemented as an AA. It supports heterogeneous receivers through the combination of layered multicast transmission with the selective filtering and pruning of layers within the active nodes. In previous work [28] a case study was conducted, which led to an initial version of ALMA. In this paper we refine the protocol and present additional results.

3.1 Requirements and assumptions

We would like to create a model for adaptive applications, that can benefit from the facility of retrieving the necessary information about network conditions using the router support that can be provided by an active network. It is well known that feedback information for congestion control must be used with care, in order to avoid generating extra packets that might worsen congestion. It is also known that state and processing in the active nodes might be expensive, and therefore should be reduced to the minimum necessary. We set these two principles as requirements for our protocol.

Since our target is a delay-sensitive application, and for efficiency reasons, we also impose the requirement of not delaying packets inside the active routers. Additionally, the protocol should not rely on specialised schedulers that decide how to handle flows in a centralised way. The decision must be built into the sessions themselves, taking into account their preferences and resource availability information. No assumptions are made regarding the exact traffic specifications for each session.

We assume a pure AN environment in which all packets are active. It is possible to optimise it in several ways, but for the moment we would like to concentrate on the protocol functionality from a conceptual point of view. Therefore we try to be as generic as possible without imposing the constraints of a real implementation. Furthermore, we assume that all nodes are active. Although this assumption seems unrealistic, the use of an equivalent link abstraction [21] enables us to easily migrate to a mixed scenario of active and non-active nodes, while at the same time keeping a pure AN abstraction so that we do not need to worry about implementation and interoperability constraints in the design phase.

3.2 Protocol basics

We consider a layered multicast application composed of a single source and several different receivers. The source generates delay-sensitive traffic encoded in a hierarchical way such that lower layers are more important than higher layers, and the higher layers are useless without the lower layers. The generated data is carried by capsules which also contain the instructions to process it inside the active nodes along the path to the receivers. The data capsules are organised in layers according to the hierarchical position of the data they carry. This is similar to conventional layered transmission [17, 24] except that in this case, since customisable capsules are used, it is feasible to use a potentially large amount of layers. Besides that, the semantics of the relationships among the layers is built into the scheme itself, facilitating its customisation to the specific characteristics of a given application.

The multicast routing mechanism employed is a simple form of source-based sparse-mode scheme similar to the well-known AN multicast example from ANTS [25]. For simplification purposes, no group address is used. Receivers subscribe directly to the address of the source they wish to receive data from. While subscribing, a receiver specifies the subscription level it wishes to obtain, which indicates how many layers it wishes to receive, like in [2]. This requires only one type of capsule: a *subscribe* capsule, which takes two parameters: the source address and the desired subscription level. To abandon all layers, a receiver spawns a subscribe capsule carrying zero as the subscription level. The main advantage of this subscription model is that a group of streams can have its semantics understood inside the network nodes as layers belonging to the same session. Besides that, such a simple subscription model allows an arbitrary number of layers to be used, since the amount of multicast forwarding state left in the nodes does not increase with the number of layers, as opposed to what would happen in conventional layering schemes using IP multicast. The disadvantage is that the processing of a subscription request in a node is more complex, since it must compute the upstream subscription level for a session, which is the maximum of the subscription levels at each outgoing interface that has members of the session.

Complete end-to-end congestion control is achieved by a combination of several efforts: *(i)* link outgoing interfaces export a *price* that is calculated as a function of the link characteristics and current load. *(ii)* data capsules filter themselves by comparing the current link price with their own *budget*; *(iii)* data capsules prune multicast tree branches affected by persistent congestion; *(iv)* receivers spawn subscribe capsules to probe for additional bandwidth.

3.3 Filtering mechanism

Link outgoing interfaces export an instantaneous price for the usage of the link. The price is calculated as a function of the link characteristics and current total link load (of all traffic traversing the link). The price in this case is no more than an indication of the level of congestion at a given interface.

The budget assigned to a data capsule is directly proportional to the marginal utility of adding its layer, which in our case expresses the relative importance of its layer with respect to the other layers. Since the marginal utility for adding an individual layer decreases as we go towards higher layers, the budget of each capsule decreases too. Therefore, a higher layer capsule is more likely to discard itself due to a rise in link prices than a lower layer capsule. The resulting behaviour is similar to a priority dropping mechanism, or a weighted RED [8]. Such a simple filtering mechanism does not require specialised schedulers, it only requires the link to export a price function that varies as a function of the load on the link.

Figure 1 shows an example of budget function, used to calculate the budget to be assigned to each layer, and an example of price function for a point-to-point link, inspired from [23]. These are the ones we used in the simulations described in Sect. 4.

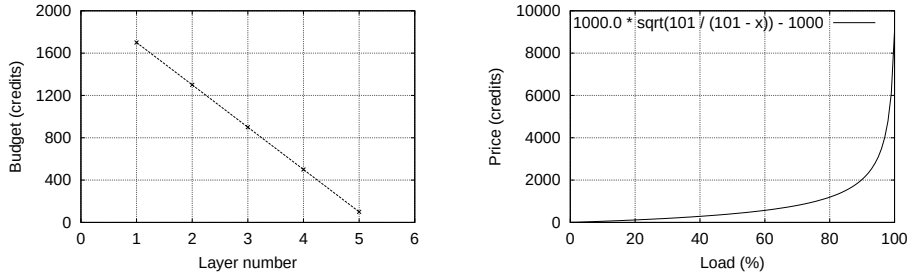

Fig. 1. The budget function to be applied to each layer (left), and the price function maintained by the node interfaces (right).

The reason to export a price and not the link load directly is that an abstract price function gives more freedom for the network managers to adjust its shape to offer different incentives to the users. This can work in a transparent way with different kinds of links. For example, in a shared Ethernet link the local interface load is not a good indication of the actual link utilisation, therefore another parameter should be used, such as the number of collisions.

Such filtering is performed at the time granularity of a capsule, therefore it must be based either on instantaneous or short term average values (i.e. of the order of a few capsules).

In the simulations presented in this paper we use instantaneous values because it is simple and completely eliminates packet drops at the outgoing links. Indeed we confirmed that during the simulations there were no packet drops due to queue overflow at the router output queues. All the drops were due to the selective discarding mechanism built into the data capsules. The problem with this approach is that large bursts are penalised. Indeed we observed that a steep

transition from a very fast link to a very slow one sometimes prevents sessions from achieving full link utilisation.

3.4 Subscription mechanism

Subscribe capsules are used by the session receivers to indicate the desired subscription level (probing), and by the active routers to control congestion (pruning).

Pruning. When long-term congestion is detected at an outgoing interface of a node, the application itself (through its data capsules running in the router) decreases its subscription level for that interface by injecting a subscribe capsule with a lower subscription level into the local Execution Environment (EE). The subscribe capsule is processed as if it had come from the congested outgoing interface. It might cause the traffic to be pruned only for the concerned interface, but if no other interfaces are requesting the pruned levels, the subscription capsule travels upstream, pruning the tree at the level of the maximum subscription level required locally.

One of the difficulties here is to determine exactly when to prune. Several criteria are possible, but we would like to find one that does not require additional state to be kept in the active nodes. We cannot prune too fast, in order to allow some bursts to go through. And we cannot delay the pruning decision too long, in order to minimise resource waste. Ideally we should be able to prune in less than a round-trip time to the nearest receiver, but this information is difficult to obtain, and would need additional per-session state. Instead, we rely on the average prices, calculated using a weighted moving average. The average price can be calculated and exported by the outgoing link interfaces, in the same way as link statistics are kept for network management purposes.

Note that the state of the multicast tree downstream from the pruned branch is not immediately updated following the prune, since the subscribe capsule issued (if any) travels only upstream. In order to ensure that any stale subscription state is appropriately refreshed, each receiver periodically spawns subscribe capsules containing the minimum subscription level experienced along the path (this information is reported by the arriving data capsules). In the interval between a prune and a refresh, subscribe capsules containing the pruned level might be ignored. This might have the positive side effect of temporarily blocking requests for a layer that has just been eliminated, but the negative effect of delaying the probe process (leading to a slower adaptation).

Probing. Since a capsule executing inside an active node has no means of knowing whether there are downstream nodes still interested in a layer which was pruned long ago due to congestion, the decision to probe for bandwidth is always taken by the receivers. To be able to probe, each receiver needs to monitor the number of layers it effectively receives, which might be different from its desired subscription level (since layers might have been pruned by the

upstream routers due to congestion). After a period in which the quality achieved is considered good with respect to the effective number of layers received, such that this number is inferior to the desired one, a receiver decides to probe for bandwidth which might eventually be available. To do that it sends a subscribe capsule increasing its subscription level by n layers. Currently n is set to one for simplicity, although it is also possible (and better) to calculated it using the available price and budget information at the receivers.

The decision of when to probe is a very difficult one. An ideal probe criterion should virtually eliminate useless probes, while at the same time allow a relatively fast reaction to grab available resources. These two goals are conflicting and therefore a compromise must be found. In the simulations we show in this paper, the probe criterion used is a combination of conditions: (i) A minimum interval between consecutive probes must be kept. (ii) The budget for the new subscription level desired must be greater than the average price observed. (iii) Prices must be stable enough, i.e. the receiver should observe a price variation under a certain threshold during an observation period, before probing. (iv) The maximum subscription level received must stay unchanged during the observation period. (v) There should be no packet losses during the observation period. In fact this last condition is redundant when all nodes are active, because when the prices are stable enough there are no losses either.

When a subscribe capsule arrives at a node, it first checks the router interface from where it came: if this interface shows persistent congestion (according to the same criterion used to prune congested branches), the subscribe capsule simply decides to terminate its execution. This results in a loose form of "self-admission control" which filters out potentially harmful probe requests. In order to avoid denial of service for new flows, this check is only performed when multicast routing state is already present for the source on the concerned interface. This naive filtering does not take into account the actual new bandwidth introduced into the system when a probe is accepted, therefore does not guarantee that an accepted probe will not cause congestion. This mechanism could be improved, for example by using explicit traffic information in probes (e.g. peak, average bandwidth). For the moment, however, we try to keep it simple and open for different types of traffic envelopes.

3.5 Summary

To summarise, in ALMA the decision to prune is always taken inside intermediate active nodes (by data or subscribe capsules), while the decision to graft (probe) is always taken by the receivers. This is a way of distributing the decision load among AN nodes, such that the right operation is performed where the data is available. As a result, in the active nodes the reaction to congestion is fast enough, using the combination of data filtering (for short term congestion) and tree pruning (for persistent congestion).

Probing for available bandwidth is a task assigned exclusively to the receivers, therefore some delay can be expected from the moment when some bandwidth is released to the moment when new flows start to use it. The duration of this delay

is a trade-off between how fast a reaction is desired and the amount of resources (bandwidth and active processing) we are ready to spend in the probing process. In the active routers, we try to minimise the impact of probes on the downstream congestion level by ignoring probe requests during congestion periods.

The feedback information is carefully used such that no extra packets are generated in a situation of congestion. The amount of state kept in each active node is of the order of the state necessary to maintain one IP multicast group. The scheme is therefore as scalable as any other multicast-based traditional scheme.

4 Simulations

We simulated ALMA in order to study its feasibility from the point of view of the adaptation to the available bandwidth and the competition among different sessions.

First of all, we designed and implemented a simplified EE and a NodeOS module over the NS simulator [19] in order to be able to simulate the execution of active packets in the NS network nodes. The simulated EE is based on the execution of capsules written in TCL language.

The percentage of queue occupancy is used to indicate the load on the link. The link prices are calculated from the instantaneous and average queue lengths at each outgoing interface. For the average queue length, a modified RED [8] queue is used. The RED queue is inactive (i.e. the maximum and minimum thresholds are set to the end of the queue) so that it behaves like a FIFO queue with a drop tail policy, and it is only used to compute the average queue length. This average is computed as an exponential weighted moving average (EWMA) [8], with its weight factor adjusted to achieve the desired convergence time. The RED module was also modified to compute the average when the packet leaves the queue, instead of when the packet enters the queue. This is because we want the rhythm of updates to be regular and proportional to the link speed instead of the input rate.

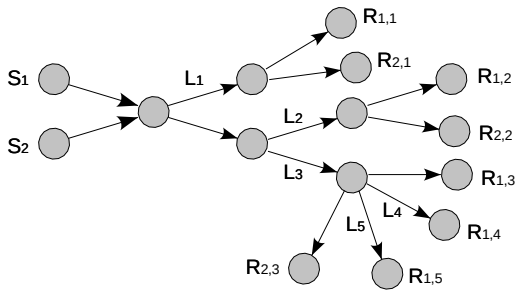

Fig. 2. Topology used in the simulations.

The topology for the simulations is depicted in Fig. 2. All nodes are active. Links L_1, L_2 and L_3 are bottlenecks with capacity 1.6 Mbps, 1.0 Mbps and 0.8 Mbps, respectively. The other links have a capacity of 4.0 Mbps each. Links L_4 and L_5 have propagation delays of 100ms and 500ms respectively, while all the other links have 10ms. Every link has an inactive RED queue of 20 packets each. Each of the sources s_1 and s_2 generates a stream with an average rate of 2 Mbps divided into 5 layers of equal average rate. The packet size is 500 bytes for both sessions. Receivers $r_{1,j}$ $(j = 1..5)$ subscribe to the 5 layers of s_1 at around $t = 1s$, while receivers $r_{2,k}$ $(k = 1..3)$ subscribe to s_2 after about one third of the total simulation time (i.e. around $t = 20/3 \approx 6.6s$), and leave the session the same amount of time later ($t \approx 13.3s$). The receivers are not synchronised, therefore they join the session at slightly different times.

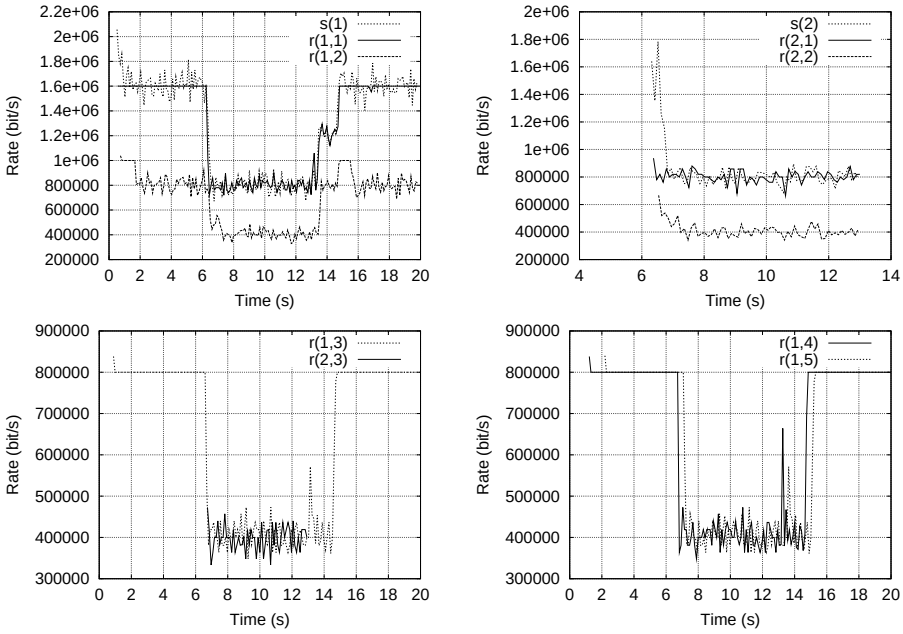

Fig. 3. Evolution of the rates in time for the members of sessions 1 and 2.

Figure 3 shows the evolution of the rates in time for the first and second sessions. We observe that after about one second the receivers get only the amount of bandwidth that fits into the bottleneck links, and the sources stop sending the exceeding amount of traffic after about one second. When the second session starts, the system accommodates it, and after about a second the two sessions are using virtually the same amount of bandwidth. We also see that when the second session terminates, the receivers of the first session are able to reuse the released bandwidth after a couple of seconds. Note that receivers $r_{1,4}$

and $r_{1,5}$ have a behaviour which is similar to $r_{1,3}$, indicating that the additional delay they experience has little impact on them. It is also worth noting that link L_2 has a capacity for a non-integer number of layers, but the receivers connected to it behave similarly to the ones connected to L_3.

We now take a closer look at the filtering behaviour, taking two of the receivers as examples. Figure 4 shows the received rate per layer at receivers $r_{1,1}$ and $r_{1,2}$. Each of the curves $Layer_i$ represents the rate of $Layer_i$ plus the cumulative rate of the lower layers.

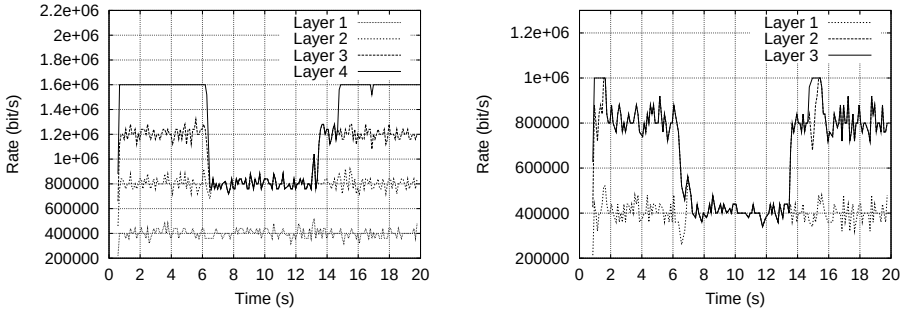

Fig. 4. Throughput per layer (cumulative), for receivers $r_{1,1}$ (left) and $r_{1,2}$ (right).

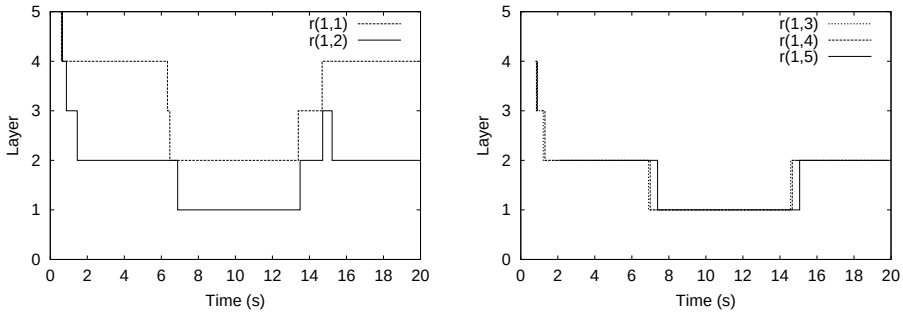

Fig. 5. Subscription levels observed at receivers $r_{1,1}$ to $r_{1,5}$

In Fig. 4, the highest layers disappear after about a second. When the second session starts, the receivers lose some more layers, but are able to get them back a couple of seconds after the second session terminates. Note the asymmetry between prunes and probes. While several layers might be pruned at once, only one layer is added at a time, during the probe period. This asymmetric behaviour can also be observed in Fig. 5 (left), which shows the subscription levels observed by the same pair of receivers. We see that dropping layers is fast while

adding them takes a longer time. Figure 5 (left) also shows that $r_{1,2}$ performs an unsuccessful probe at $t \approx 15s$. In Fig. 5 (right) we can see that the effect of the delay on the subscription levels observed at receivers $r_{1,3}$, $r_{1,4}$, and $r_{1,5}$ is relatively small, confirming what Fig. 3 also shows us.

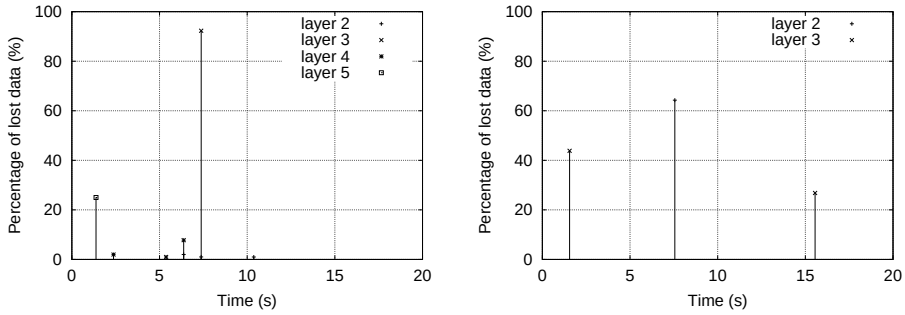

Fig. 6. Percentage of data capsules discarded for $r_{1,1}$ (left) and $r_{1,2}$ (right).

Figure 6 shows the filtering behaviour for each layer, again taking $r_{1,1}$ and $r_{1,2}$ as samples. In this figure, the percentage of data capsules lost, as calculated by each of the receivers over an interval of one second, is plotted. Since queue overflow does not occur, and link errors are not simulated, all the losses plotted correspond to filtered packets. If we compare with Figure 5, we can see that the peaks of loss that a given layer suffers correspond to instants immediately before the layer is pruned. Note that the losses observed at $t \approx 15.5s$ are due to the unsuccessful probe previously sent by $r_{1,2}$ (observed in Fig. 5 (left)).

5 Conclusions and Future Work

We presented an active multicast adaptation protocol that addresses receiver heterogeneity not only by performing selective filtering on data packets but also by pruning unused multicast tree branches and probing for available bandwidth. The protocol uses source-based reverse path routing, does not need multicast groups, and is based on hierarchical layers that are built into its capsule types. It is independent on scheduling algorithms - a simple FIFO queueing is enough - but counts on the ability of the active nodes to export link parameters that can be used in the adaptation process.

The protocol has shown to be a feasible AA, and it motivates us to pursue further research in this direction. The next immediate steps are the simulation and implementation of the mechanisms described over more complex and realistic AN environments consisting of multiple node and link types, as well as in a fully heterogenous network in which some of the routers are active while others are not. Other topics for future research include: improve the reactiveness of ALMA in scenarios presenting multiple short duration sessions; cover routing issues such

as the scalability to a large number of sessions with few receivers; trade several types of resources; address budget assignment and management issues within a session; dynamically deploy price functions.

The task of performing adaptation did not turn out to be trivial even when using all the AN support we could imagine. Basically the same trade-offs of classical control algorithms remain, such as stability versus reaction time, feedback availability versus bandwidth consumption, etc. On the other hand, we can easily choose where to place a given functionality (e.g. pruning of unused tree branches), and therefore try to place it as close as possible to the data it needs (in this article, congestion information). The resulting protocol is able to share the decision load among all active nodes involved in the session, with most of the complexity still at the receivers, due to the task of demultiplexing and decoding. The amount of state and processing in the active nodes is kept as small as possible, and does not increase with the number of layers used.

As a side effect, a slightly different model of active node pops up: a model in which the intelligence is distributed among the active applications representing the user interests, and the active resource managers representing the network provider interests. These different types of agents cooperate to seek configurations which are both locally and globally optimal with respect to the resources used, user satisfaction and provider revenues. In [27] we have developed this model further, and we are now redesigning ALMA based on those results.

6 Acknowledgements

This work has been carried out within the TINTIN project funded by the Walloon region in the framework of the programme *"Du numérique au multimédia"*.

References

1. S. Athuraliya, D. Lapsley, S. Low, "An Enhanced Random Early Marking Algorithm for Internet Flow Control", Proc. INFOCOM'2000, Tel-Aviv, Israel, March 2000.
2. A. Banchs et al., "Multicasting Multimedia Streams with Active Networks", Proc. IEEE Local Computer Networks Conference, LCN'98, Boston, USA, October 1998, pp.150-159.
3. B. Cain, T. Speakman, D. Towsley, "Generic Router Assist (GRA) Building Block Motivation and Architecture", IETF RMT Working Group, Internet draft, March 2000, work in progress.
4. A.T. Campbell et al., "A Survey of Programmable Networks", ACM SIGCOMM Computer Communication Review, April 1999, p.7-23.
5. Y. Chae et al., "Exposing the Network: Support for Topology-sensitive Applications", Proc. OPENARCH'2000, Tel-Aviv, Israel, March 2000.
6. T. Faber, "ACC: Using Active Networking to Enhance Feedback Congestion Control Mechanisms", IEEE Network (Special Issue on Active and Programmable Networks), May/June 1998, p.61-65.
7. D. F. Ferguson et al., "Economic Models for Allocating Resources in Computer Systems", in Market-Based Control: A Paradigm for Distributed Resource Allocation, Scott Clearwater (ed.), World Scientific Press, 1996.

8. S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, August 1993.
9. S.J. Golestani, S. Bhattacharyya, "A Class of End-to-End Congestion Control Algorithms for the Internet", Proc. ICNP, October 1998.
10. R. Gopalakrishnan et al., "A Simple Loss Differentiation Approach to Layered Multicast", Proc. INFOCOM'2000, Tel-Aviv, Israel, March 2000.
11. K. Jun et al., "Intelligent QoS Support for an Adaptive Video Service", Proc. IRMA 2000.
12. S.K. Kasera et al., "Scalable Fair Reliable Multicast Using Active Services", IEEE Network (Special Issue on Multicast), January/February 2000.
13. R. Keller et al., "An Active Router Architecture for Multicast Video Distribution", Proc. INFOCOM'2000, Tel-Aviv, Israel, March 2000.
14. F.P. Kelly, A.K. Maulloo, D.K.H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability" Journal of the Operational Research Society, vol. 49 issue 3, pp.237-252, March 1998.
15. S. Low, D.E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence", IEEE/ACM Transactions on Networking, 1999.
16. M. Luby, L. Vicisano, T. Speakman, "Heterogeneous multicast congestion control based on router packet filtering", (work in progress), RMT working group, June 1999, Pisa, Italy.
17. S. McCanne, V. Jacobson, M. Vetterli, "Receiver-driven layered multicast", in *Proc. ACM Sigcomm*, pages 117-130, Palo Alto, California, August 1996.
18. K. Najafi, A. Leon-Garcia, "A Novel Cost Model for Active Networks", Proc. Int. Conf. on Communication Technologies, World Computer Congress 2000.
19. UCB/LBNL/VINT Network Simulator - ns (version 2), *http://www-mash.cs.berkeley.edu/ns/*.
20. S. Sarkar, L. Tassiulas, "Fair Allocation of Discrete Bandwidth Layers in Multicast Networks", Proc. INFOCOM'2000, Tel-Aviv, Israel, March 2000.
21. R. Sivakumar, S. Han, V. Bharghavan "A Scalable Architecture for Active Networks", Proc. OPENARCH'2000, Tel-Aviv, Israel, March 2000.
22. D. L. Tennenhouse et al., "A Survey of Active Network Research", IEEE Communications Magazine, Vol. 35, No. 1, pp80-86. January 1997.
23. C. Tschudin, "Open Resource Allocation for Mobile Code", Proc. the Mobile Agent'97 Workshop, Berlin, Germany, April 1997.
24. L. Vicisano, J. Crowcroft, L. Rizzo, "TCP-like congestion control for layered multicast data transfer", Proc. IEEE INFOCOM'98, San Francisco, USA, March/April 1998.
25. D. J. Wetherall, J. V. Guttag, D. L. Tennenhouse, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols", Proc. OPENARCH'98, San Francisco, USA, April 1998.
26. H. Yamaki, M.P. Wellman, T. Ishida, "A market-based approach to allocating QoS for multimedia applications", Proceeding of the Second International Conference on Multiagent Systems (ICMAS-96), Kyoto, Japan, December 1996.
27. L. Yamamoto, G. Leduc, "An Agent-Inspired Active Network Resource Trading Model Applied to Congestion Control", Proc. MATA'2000 Workshop, Paris, France, September 2000.
28. L. Yamamoto, G. Leduc, "Adaptive Applications over Active Networks: Case Study on Layered Multicast", Proc. ECUMN'2000, Colmar, France, October 2000.