Proc. 1st IEEE European Conference on Universal Multiservice Networks (ECUMN 2000), Colmar, France, October 2000.

Adaptive Applications over Active Networks: Case Study on Layered Multicast

Lidia Yamamoto, Guy Leduc

Research Unit in Networking, University of Liège
Institut Montefiore, B28, B-4000 Liège, Belgium
Tel.: +32 4 366 2766, +32 4 366 26 98
Fax.: +32 4 366 29 89
Email: yamamoto@run.montefiore.ulg.ac.be,
leduc@montefiore.ulg.ac.be

Abstract

In this paper we study the potential and limitations of active networks in the context of adaptive applications. We present a survey of active networking research applied to adaptive applications, and a case study on a layered multicast active application. This active application is a congestion control protocol that selectively discards data in the active routers, and prunes multicast tree branches affected by persistent congestion. Our first results indicate that active networks can indeed help such an application to adapt to heterogeneous receivers, with a minimum amount of state overhead, equivalent to that of a single IP multicast group.

Keywords: Active network, multicast, adaptive application, congestion control.

1. Introduction

The need for a network technology able to transport multiple existing and emerging services was detected long ago, and is now reinforced by the convergence of computing and communications, and by the variety of available fixed and mobile network technologies. Earlier attempts such as ISDN, ATM-based B-ISDN and the enhancement of IP for QoS support require long years of standardisation and therefore lack the flexibility to quickly adapt to a wide range of service characteristics and requirements over a wide range of transmission technologies, in a rapidly changing network environment. Recent advances in mobile software and distributed systems are opening a new path towards active and programmable networks [1, 2, 3]. Active networks (AN) allow the network managers or users to program the network nodes according to their needs, offering a great amount of flexibility. They become therefore an ideal candidate for complementing existing networks towards a truly multiservice environment, that can quickly adapt to new service requirements and transmission technologies.

Active networks bring a lot of promises, such as to enhance existing networks with customisable services, that would enable network operators and service providers to offer user-driven instead of manufacturer-driven solutions. However, few active applications seem to be available at the moment, which can really show the benefits and trade-offs involved. For the moment most of the efforts are concentrated on AN platform architectures, with relatively few applications which are not just usage examples of a given AN platform. We believe that building more applications can help understanding the requirements on new AN platforms, and therefore help refining these platforms.

Of course it is still too early to say what the future of active networking technology will be. There is still room for a lot of research in crucial issues such as security, performance, resource management and others. The focus of our work is on resource management from the point of view of an adaptive application. Such an application must be able to react to changing network conditions in a very dynamic way in order to adapt to the existing resource availability and its possible fluctuations.

In this paper we study the potential and limitations of active networks in the context of adaptive applications. We present a survey of active networking research applied to such applications, and a case study on a layered multicast active application.

2. Background

In this section we give an overview of adaptive applications in the context of active networks. We start with a brief summary of what active networks are, the current architectures proposed and environments available. Then we turn to multimedia adaptation using active networks, focusing on the special case of layered multicast, which is the subject of our study.

2.1. Active Networks

Downloading mobile code is already a routine over the Internet, such as the use of Java applets and mobile agent platforms. But mobile code technology can also be applied to computer networks in order to accelerate the deployment of new protocols and services. The nodes of an active network [1] are capable not only of forwarding packets as usual but also to load and execute mobile code. The code can be transported by specialised signalling channels (programmable networks) or within special packets called "capsules" (active networks). Capsules might contain the code itself (such as in [4]) or a reference to it, such that it can be downloaded when the first capsule containing the reference arrives at a given node (such as in [5]). This allows us to program the nodes in a much more flexible and dynamic way than with the current network technologies such as IP or ATM. Within an active or programmable router, it is possible to load a new protocol or service, and to remove protocols or services that are no longer useful, without having to shut down the router, thus without service interruption. If the distinction between active and programmable networks seemed at some point in time clear [6], the tendency today seems to be towards an integration of the two concepts, since both are forms of achieving open programmability in networks [3], and special flavours in between or combining both approaches are also possible [7].

A framework for an active node architecture is being proposed within the AN community [8]. It includes a supporting operating system (the NodeOS), one or more execution environments (EE), and the active applications (AA). The NodeOS is responsible for managing local resources such as CPU processing time, link bandwidth and memory storage. On top of the NodeOS, a number of EEs can be installed. On top of each EE, various AAs can be dynamically loaded and executed. The EE is responsible for controlling the access from the AAs to local resources, and limiting resource usage depending on specified policies.

The NodeOS plays a crucial role in providing access to local node resources, as well as information about resource availability. A NodeOS API is currently being defined [9]. At the moment this API treats four types of resources: computation, memory, communication, and persistent storage. The communication resource is handled through the channel abstraction, which when ready should include QoS support, as well as access to link information such as bandwidth, queue length, and other properties and statistics.

When not all the network nodes are active, it is necessary to discover resources outside an active node. For this purpose, complementary efforts such as CMU Remos [10] could be used. The CMU Remos interface enables network-aware applications to obtain network properties such as topology, latency and bandwidth. Another interesting approach appears in [11], where an *equivalent link* abstraction is proposed, such that from an AN point of view it is possible to look at a set of non active nodes as a single link, with some mechanisms needed to discover the (possibly changing) properties of such a virtual link.

2.2. Adaptive Applications

Adaptive applications can tolerate fluctuations in resource availability, and are necessary in a heterogeneous environment such as the Internet today, where different network technologies and user terminals are interconnected together, and over which a multitude of services coexist. In the case of multimedia applications, a good survey can be found in [12]. The adaptation mechanisms can be implemented at several layers of the protocol stack, ranging from pure application layer techniques to network level protocols. For example, we can adapt to the available bandwidth using elastic traffic that reduces the data rate generated in presence of network congestion. Fluctuations in delay can be dealt with by using elastic buffers to adjust the play-out time. To deal with CPU and memory bottlenecks, some interaction with the operating system is necessary (see Section 5 of [12] for examples). Our focus in this paper is on network and transport level mechanisms for adaptation.

One of the main difficulties encountered in classical adaptation approaches is how to obtain the required information about resource availability, mainly when this information is hidden in a blackbox network and has to be inferred using only some indirect indications that are observed at the end systems. Using active networks, new models for adaptive applications could be envisaged, which can benefit from the possibility to send mobile code or agents to certain elements inside the network. These agents can be in charge of collecting information about network conditions, without having to rely on indirect indications or on heavy signalling protocols. Indeed, the idea of sending small pieces of code directly to where the data needs to be treated, instead of exchanging a large amount of data, is one of the main motivations of mobile agent technology, and it can also be applied to mobile code in the case of active networks.

Actually many adaptation mechanisms come from the world of mobile agents. Some examples are: In [13] an open resource allocation scheme based on market models is applied to the case of memory allocation for mobile code. In [14] 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 [15] 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.

Several adaptive algorithms inspired on operational research and economy theories have been proposed to control resource usage in networks. These algorithms are able to converge towards a globally optimal resource allocation in a decentralised way. For an overview on the topic see [16]. In [17, 18, 19] such theories are applied to the problem of end-to-end congestion control (i.e. where bandwidth is the scarce resource). In [17] 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 [18] 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 [19] a similar algorithm is proposed, and in a more recent work [20] 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 our case study, is not straightforward due to fairness issues, as pointed out in [21]. A cost model for active networks is proposed in [22], which expresses the trade-off between different types of resources in a quantitative way. However, the recursive approach adopted makes its usage for multicast applications probably impractical.

A lot of AN example applications are related to information filtering or transcoding in the active nodes, as a form of adaptation to bandwidth limitations. In [23], a video plug-in designed for multicast video distribution is built over a high performance active node platform. The video plug-in is loaded on-demand using a code caching scheme, and runs in kernel space in order to achieve high performance. The video stream is encoded using a highly scalable codec. When congestion occurs, the plug-in performs fine-grain selective dropping of video packets containing higher frequency coefficients, which carry image details and are not crucial for the image definition. The use of the video plug-in shows significant improvement in video quality under congestion, with respect to the situation without the plug-in. Other examples of application-tailored selective discard modules implemented over an active network platform can be found in [11, 7]. The common idea to these examples is to show how more intelligent functionality in the routers can help improving the reception quality of a multimedia stream.

2.3. Layered Multicast

A special class of adaptive applications is layered multicast, in which the problem of multicasting to heterogeneous receivers, under heterogeneous network conditions, is dealt with by using a hierarchical encoding scheme, in which the data stream is divided into a number of layers, such that the lower layers contain the basic information, and the upper layers successively refine this information in order to obtain a higher quality when enough bandwidth is available. An example of such an encoding scheme can be found in [23] as mentioned in Section 2.2.

In a pure end-to-end layered multicast scheme [24, 25], 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 existing end-to-end layered schemes suffer from the limitations of pure best-effort networks which make most of the Internet today, and therefore present many problems: 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 [26, 27, 28], that can count on router support in order to solve the abovementioned problems. However, 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 multicast sessions are heterogeneous by nature, 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 that are loaded on-demand, and let them evolve through usage experience.

We cited some examples in Section 2.2, that selectively filter information inside the active nodes in order to provide an improved delivery quality under bandwidth constraints. In these examples, the source of congestion does not react in order to prevent overflow and discarding. Intelligent filtering in the intermediate active nodes alleviates downstream congestion while preserving most of the quality. It 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. Case study on layered multicast

This section shows a case study in the domain of layered multicast. The goal of this study is to provide a sample AN application that enables a better understanding of the trade-offs involved with the AN paradigm, mainly with respect to resource management. For the moment we only consider bandwidth as a monitored resource, making our active application essentially a multicast congestion control scheme.

We consider a layered multicast application composed of a single source and several different receivers. The source generates real-time delay sensitive traffic encoded in a hierarchical way such that the information carried in lower layers is more important than the one in higher layers. Additionally, data from higher layers might depend on data from lower layers to be decoded. The hierarchical stream is carried by capsules which also contain instructions about the behaviour to adopt in face of congestion 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 [24, 25] 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 which is similar to typical AN usage examples such as [5]. 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. 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.

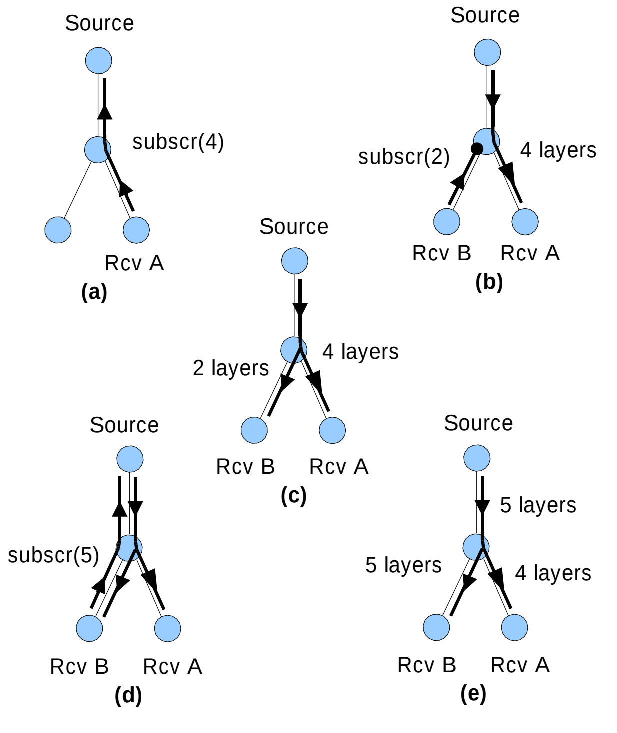

Figure 1: Subscribe mechanism in different situations.

Figure 1 illustrates the behaviour of the subscribe mechanism. When the session is empty (a) no traffic flows through the network. When the first receiver ("rcv A" in the figure) joins the session requesting 4 layers, its subscribe request travels upstream until it reaches the source. The source then starts sending 4 layers of data (b). Then a second receiver joins with 2 layers ("rcv B" in (b)). Its subscription request stops in the active node, since it is inferior to the current subscription level coming from the upstream router (in this example, the source itself). The data capsules for the first two layers now duplicate themselves in the directions of receivers A and B (part (c) of figure). In (d), the receiver now requests 5 layers. Its subscription capsule now reaches the source, which starts sending an additional layer (e).

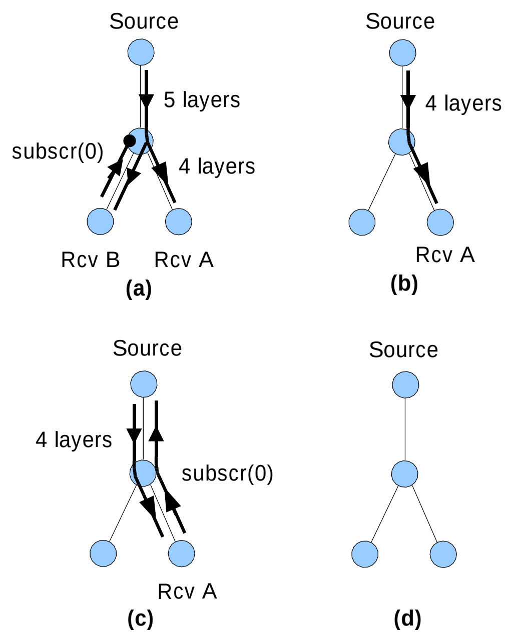

Figure 2: Subscribe mechanism in different situations.

Figure 2 explains the unsubscribe mechanism in a similar way. Initially (2(a)), receiver B is getting 5 layers, resuming the situation sketched in 1(e). When receiver B wishes to leave the session, it issues a subscribe capsule with zero as the requested number of layers (still in 2(a). Since receiver A is still participating in the session, the subscribe request stops at the active router, which also prunes the tree branch leading to B (part (b) of figure). When receiver A also decides to leave, its subscribe request will travel to the source (c), and the session will become empty, with no data capsules flowing any longer.

This simple subscription method allows any number of layers to be used, since the amount of state left in the nodes does not increase with the number of layers (as it 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 has to keep track of the maximum subscription levels at each outgoing interface in order to compute the upstream subscription level.

From an abstract point of view, all capsules can be regarded as small mobile agents with a certain degree of autonomy, much in the spirit of *messengers* [29]. They travel to network nodes where they decide when to continue or stop the trip (e.g. stop due to congestion), and when to fork new capsules (in a multicast branch).

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)* subscribe capsules probe for additional bandwidth.

This is a pure AN approach in which all capsules carry code. It is possible to optimise it in several ways [7, 30], but for the moment we are not concerned with the actual performance but only with the necessary functionality. 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 [11] 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.

The budget assigned to a data capsule is directly proportional to the relative utility of its layer, which in our case expresses its relative importance with respect to the other layers. Since the relative utility of 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 [31]. Such a simple filtering mechanism does not require specialised schedulers, it only requires that the link exports a price function that varies as a function of the load on the link. The reason why not exporting the load directly is that an abstract price function gives more freedom to adapt its shape to offer different incentives to the users.

When long-term congestion is detected at an output interface of a node, the application itself (through its data capsules running in the router) decreases its subscription level for that interface by issuing a subscribe capsule with a lower subscription level. The subscribe capsule is injected into the local EE and processed as if it had come from the congested outgoing interface. It will first prune the traffic on the concerned interface, and then, if no other interfaces are requesting the pruned levels, it will travel upstream, pruning the tree at the level of the maximum subscription level required locally.

Since a capsule executing inside an active node has no means to know if 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, but 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 one layer.

When a subscribe capsules 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 bandwidth 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 clearly be improved, but for the moment we try to keep it as simple as possible.

To summarise, the decision to prune is always taken inside intermediate active nodes (by data or subscribe capsules), while the decision to graft is always taken by the receivers. This is a way of distributing the decision load among AN nodes, such that an operation is performed at the place where the data it needs is available. As a result, in the active nodes the reaction to congestion is fast, 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 amount of such delay is a trade-off between how fast a reaction is desired and how much bandwidth we are ready to spend in the probing process. 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. Results

We simulated our AN layered multicast application in order to study its feasibility from the point of view of 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 [32] 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.

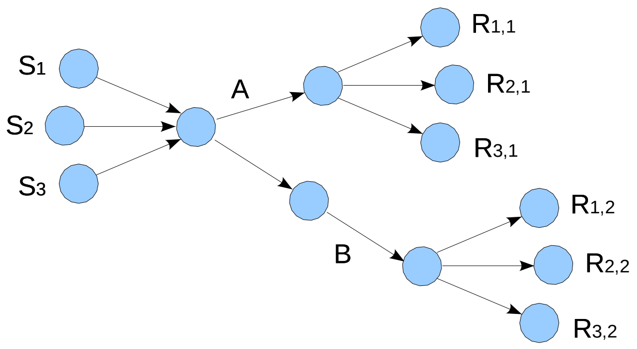

Figure 3: Topology used in the simulation.

The topology used in the simulation is illustrated in Figure 3. All nodes are active. All links are point-to-point bidirectional and symmetric, with a propagation delay of 10 ms in both directions, and a drop-tail FIFO queue. Links A and B are bottlenecks with bandwidths of 1200 kbps and 800 kbps, respectively. The other links have the capacity of 1 Mbps each.

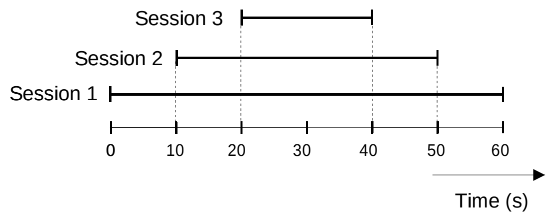

Figure 4: Lifetimes of the sessions involved in the simulation.

The simulation includes three sessions of one source and two receivers each: session i ($i = 1..3$) is comprised of source s_i and receivers $r_{i,j}$ ($j = 1..2$). Each of the sources s_i generates a stream with an average rate of 1 Mbps divided into 5 layers of equal average rate.

Receivers $r_{1,j}$ subscribe to the 5 layers of s_1 at the beginning of the simulation, while receivers $r_{2,j}$ and $r_{3,j}$ subscribe to their respective sources after 10 seconds and 20 seconds of simulation time, respectively, and leave the session at $t \approx 50s$ and $t \approx 40s$, respectively. This can be better explained through Figure 4, which indicates the lifetimes of each session. The purpose of this join and leave order is to show the reaction to arriving and leaving sessions, and it is symmetric in order to facilitate the visualisation of the results.

Figure 5 show the evolution of the rates in time for the first

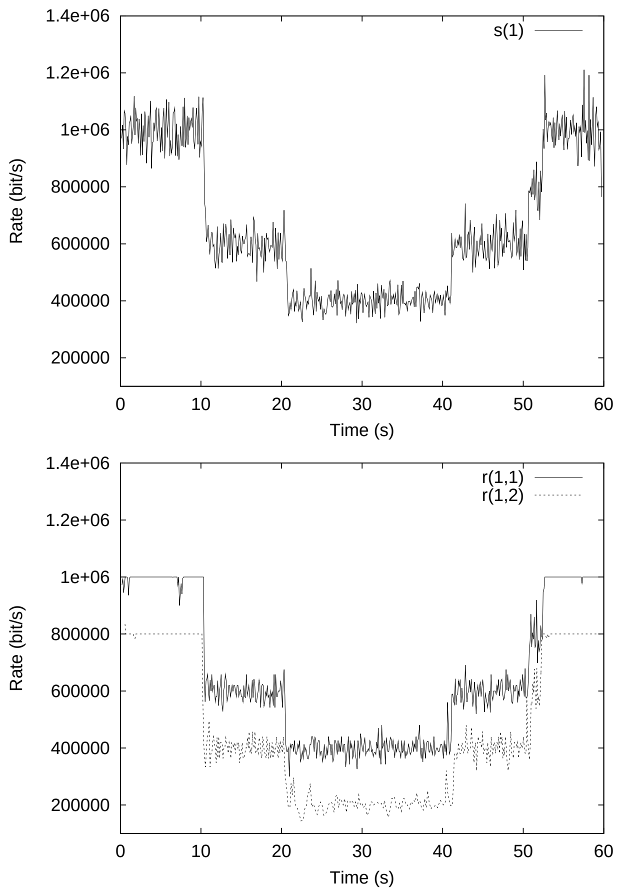

Figure 5: Evolution of the rates in time for the first session: source s_1, receivers $r_{1,1}$ and $r_{1,2}$.

session. At the beginning of the simulation, s_1 transmits all layers. While $r_{1,1}$ is able to receive all layers, $r_{1,2}$ receives only 4 layers, corresponding to its bottleneck bandwidth. When the second and third sessions start, the first session progressively accommodates itself to the new situation. After the second and third sessions finish, the first session is able to resume its initial configuration. As expected, the curve for s_1 closely follows that of $r_{1,1}$, the receiver with the largest amount of available bandwidth. This shows that the prune mechanism is indeed effective in eliminating superfluous layers. The same happens to the other sessions (not shown).

The reaction to a sudden congestion situation is faster than the reaction to newly available bandwidth. This is especially visible when comparing the first 10 seconds with the last 10 seconds of simulation shown in Figure 5.

The slower probe time compared to the prune time is a result of the strict probe criteria used. While several layers may be pruned at once, only one layer may be added at a time, and only when a combination of several conditions is met: *(i)* The minimum interval between consecutive probes is set to one second. *(ii)*The layer budget for the new subscription level desired must be greater than the average price observed. *(iii)*The average prices must be stable enough or decreasing during an observation period, before probing. The observation period is set to the minimum between the time to receive 20 packets and the probe interval. *(iv)* The maximum subscription level received must stay unchanged during the observation period.

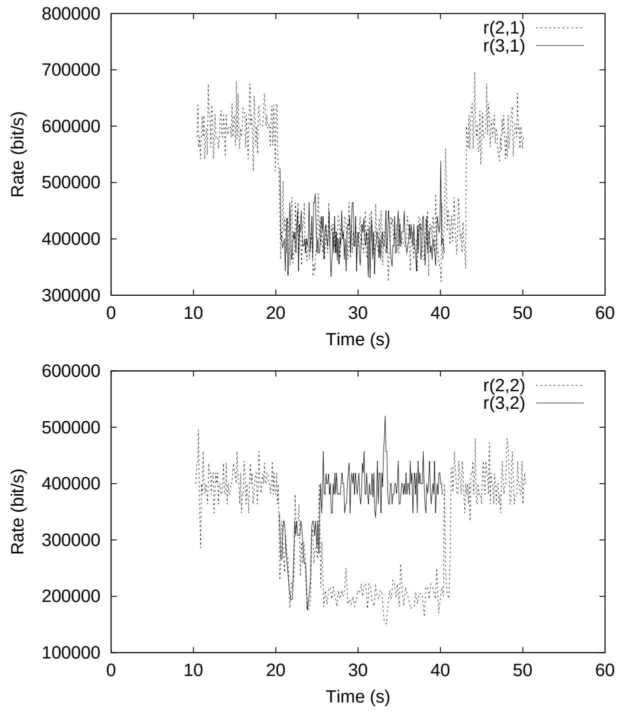

Figure 6: Evolution of the rates in time for two receivers sharing bottleneck A ($r_{2,1}$ and $r_{3,1}$, upper graphic), and two receivers sharing bottleneck B ($r_{2,1}$ and $r_{3,1}$, lower graphic).

The decision of when to probe is a very difficult one. Further studies are needed to improve the current probe method. An ideal probe rule 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.

Another difficulty is that the task of probing is delegated to the receivers, while the information for a successful probe is scattered: the information about bandwidth availability is present in the active nodes, while the receivers know about their preferences and the traffic characteristics of each layer (e.g. average bandwidth).

We now turn to the second and third sessions. Figure 6 shows that the receivers under bottleneck A equally share the available bandwidth, also when comparing to $r_{1,1}$ in Figure 5. On the other hand, the same does not happen in the case of link B: when the three sessions are active, there is enough bandwidth for around 1.33 layers. The utility of receiving one third of a layer is questionable. The result in this case is that one of the receivers ($r_{3,2}$) ends up with two layers and the others with one layer each. The fairness issues involved when dealing with discrete layers have been studied and formalised in [21], and it turns out to be a complex problem. In [21] the author proposes a centralised algorithm to allocate the layers, but how close decentralised schemes, such as the one we present in this paper, can get from the ideal solution, seems to be still an open question.

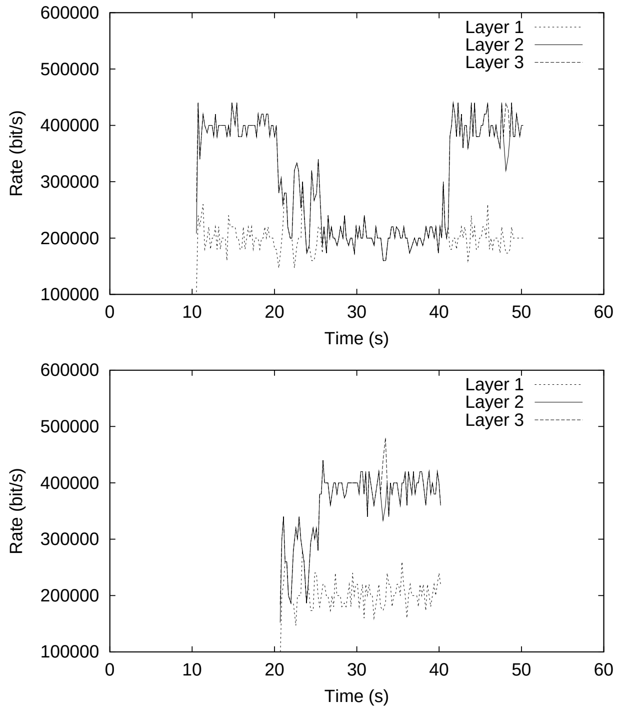

Figure 7: Cumulative layers for receivers $r_{2,2}$ and $r_{3,2}$.

It is also important to check how each individual layer is affected by congestion. Figure 7 shows the received rate per layer at receivers $r_{2,2}$ and $r_{3,2}$. Each of the curves $Layer_i$ represents the rate of $Layer_i$ plus the cumulative rate of the lower layers. This figure basically conveys the same information as shown in the lower part of figure 6, but on a per-layer basis. We can see that the base layer is essentially unaffected, showing that the filtering mechanism selectively discards data according to the importance of the layer and the network conditions.

5. Conclusions

We learned several lessons from this experience. First of all, our initial impression that it would be trivial to perform adaptation using a full AN infrastructure, was clearly wrong. Relatively complex capsules are needed, and basically the same trade-offs of classical control algorithms are encountered, such as stability versus reaction time, feedback availability versus bandwidth consumption, etc. On the other hand, we have a lot of flexibility to choose where to place a given functionality, and we can therefore try to place it as close as possible to the data it needs. For the case of pruning, this task has shown to be easy, but the same does not apply to the probe procedure.

The next step now is to perform extensive tests including a wider variety of cases, in simulations as well as on a real active network environment using multiple node and link types, as well as in a mixed network in which some of the routers are active while others are not. The performance penalty, which cannot be measured in simulations, needs to be evaluated on a real network. We are also enhancing our sample application to treat multiple sources using a shared tree.

References

[1] D. L. Tennenhouse et al., "A Survey of Active Network Research", IEEE Communications Magazine, Vol. 35, No. 1, pp80-86. January 1997.

[2] J. Smith et al., "Activating Networks: A Progress Report", IEEE Computer, April 1999, p.32-41.

[3] A.T. Campbell, et al., "A Survey of Programmable Networks", ACM SIGCOMM Computer Communication Review, April 1999, p.7-23.

[4] M. Hicks et al., "PLANet: An Active Internetwork", Proceedings of IEEE INFOCOM'99, New York, 1999.

[5] D. J. Wetherall, J. V. Guttag, D. L. Tennenhouse, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols", Proceedings of IEEE OPENARCH'98, San Francisco, USA, April 1998.

[6] T.M. Chen, A.W. Jackson, "Active and Programmable Networks", Guest Editorial, IEEE Network, May/June 1998, p.10-11.

[7] G. Hjálmtýsson, "The Pronto Platform: A Flexible Toolkit for Programming Networks using a Commodity Operating System", Proceedings of IEEE OPENARCH'2000, Tel-Aviv, Israel, March 2000, p. 98-107.

[8] K.L. Calvert (ed) et al., "Architectural Framework for Active Networks", (DARPA) AN Working Group, draft version 1.0, July 1999, work in progress.

[9] L. Peterson (ed) et al., "NodeOS Interface Specification", (DARPA) AN NodeOS Working Group, draft, January 2000, work in progress.

[10] N. Miller, P. Steenkiste, "Collecting Network Status Information for Network-Aware Applications", Proceedings of IEEE INFOCOM'2000, Tel-Aviv, Israel, March 2000.

[11] R. Sivakumar, S. Han, V. Bharghavan "A Scalable Architecture for Active Networks", Proceedings of IEEE OPENARCH'2000, Tel-Aviv, Israel, March 2000.

[12] B. Vandalore et al., "A Survey of Application Layer Techniques for Adaptive Streaming of Multimedia", to appear in the Journal of Real Time Systems, 2000.

[13] C. Tschudin, "Open Resource Allocation for Mobile Code", Proceedings of the Mobile Agent'97 Workshop, Berlin, Germany, April 1997.

[14] K. Jun, L. Bölöni, D. Yau, D.C. Marinescu, "Intelligent QoS Support for an Adaptive Video Service", To appear in the Proceedings of IRMA 2000.

[15] 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.

[16] D. F. Ferguson, C. Nikolaou, J. Sairamesh, Y. Yemini, "Economic Models for Allocating Resources in Computer Systems", Market based Control of Distributed Systems, Ed. Scott Clearwater, World Scientific Press, 1996.

[17] S.J. Golestani, S. Bhattacharyya, "A Class of End-to-End Congestion Control Algorithms for the Internet", Proceedings of ICNP, October 1998.

[18] 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.

[19] S. Low, D.E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence", IEEE/ACM Transactions on Networking, 1999.

[20] S. Athuraliya, D. Lapsley, S. Low, "An Enhanced Random Early Marking Algorithm for Internet Flow Control", Proceedings of INFOCOM'2000, Tel-Aviv, Israel, March 2000.

[21] S. Sarkar, L. Tassiulas, "Fair Allocation of Discrete Bandwidth Layers in Multicast Networks", Proceedings of IEEE INFOCOM'2000, Tel-Aviv, Israel, March 2000.

[22] K. Najafi, A. Leon-Garcia, "A Novel Cost Model for Active Networks", Proc. of Int. Conf. on Communication Technologies, World Computer Congress 2000.

[23] R. Keller et al., "An Active Router Architecture for Multicast Video Distribution", Proceedings of IEEE INFOCOM'2000, Tel-Aviv, Israel, March 2000.

[24] S. McCanne, V. Jacobson, M. Vetterli, "Receiver-driven layered multicast", Proceedings of ACM SIGCOMM, Palo Alto, USA, August 1996, p. 117-130.

[25] L. Vicisano, J. Crowcroft, L. Rizzo, “TCP-like congestion control for layered multicast data transfer”, Proceedings of IEEE INFOCOM’98, San Francisco, March/April 1998.

[26] 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.

[27] R. Gopalakrishnan et al., “A Simple Loss Differentiation Approach to Layered Multicast”, Proceedings of IEEE INFOCOM’2000, Tel-Aviv, Israel, March 2000.

[28] 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.

[29] C. Tschudin, “On the Structuring of Computer Communications”, PhD. thesis, Université de Genève, Switzerland, 1993.

[30] T. Wolf, D. Decasper, C. Tschudin, “Tags for High Performance Active Networks”, Proceedings of IEEE OPENARCH’2000, Tel-Aviv, Israel, March 2000.

[31] S. Floyd, V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance”, IEEE/ACM Transactions on Networking, August 1993.

[32] *UCB/LBNL/VINT Network Simulator - ns (version 2), http://www-mash.cs.berkeley.edu/ns/.*