# Multimedia Data Distribution and Processing in IP Networks

**Eva Hladká**


Submitted for the habilitation at
The Faculty of Informatics, Masaryk University

Brno, 2007

Except where otherwise indicated, this thesis is my own original work.

Eva Hladká
Brno, August 2007

## ACKNOWLEDGMENTS

My thanks go first and foremost to my husband, Luděk Matyska for support and hinterland for my work, for help and trust in my work.

I would also like to thank my colleagues and bachelor, master, and doctoral students in the Advanced networking technologies laboratory. They have helped me to create a very lively collaborative team. Namely thanks go to Jiří Denemark, Miloš Liška, Tomáš Rebok and Michal Procházka.

I extend a special vote of thanks to Petr Holub who collaborated most time on the partial steps on this way and was my-right hand man in all my endeavors connected with this work.

I would also like to acknowledge David Antoš especially for help on formal face of this thesis.

Finally, I would like to thank everybody using our Active Elements and the whole multimedia processing environment, thus inspiring its further development.

A more formal acknowledgment goes to the Masaryk University and its Faculty of Informatics, which provided me with the space and environment in which I can meet people and play with my ideas without any obstacles. Finally, I gratefully acknowledge CESNET, for supporting me through the Research plan "Optical Network of National Research and Its New Applications" (MŠM 6383917201) and Masaryk University for supporting me through the Research plan "Parallel and Distributed Systems" (MŠM 0021622419). The openVPN solution was inspired by my work within the EU Ithanet project (RI-2004-026539), whose support is also appreciated.

E. H.

# Contents

# Abstract

Given the importance of synchronous multimedia transmissions for collaborative environments, this thesis contains results of research on the concept of the user-empowered virtual overlay networks for synchronous multi-point distribution of high-bandwidth data. The key element for building these virtual networks is the Active Element (AE), which can be run as a stand-alone application for small collaborative environments, a network of AEs for larger groups, or even as a distributed AE on a tightly coupled cluster for handling streams with extreme bandwidth requirements. We describe models and architectures of these systems and demonstrate their performance using advanced collaborative applications. In addition to simple data distribution, we demonstrate how the AEs and the whole virtual networks can be used for data processing to support groups of unequal participants, subgrouping, and other special purposes.

# Chapter 1

# Introduction

Contemporary Internet is becoming a multipurpose transmission medium, used for multitude of applications, served previously by dedicated independent infrastructures. The ability of the Internet to facilitate collaboration leads to widespread use of various videoconferencing and more advanced collaborative environments. As a result, synchronous multimedia transmissions have become more common. Various communication patterns emerged: from many-to-many low-bandwidth streams for large scale collaboration over slow links to few-to-few extreme-bandwidth streams as seen in collaboration based on high-definition (HD) [Appendix H], [15] or even post-HD video [22]. These applications require Internet to become more active, the classical passive transmission service is no longer sufficient.

Multimedia streams are processed within the network, allowing e.g., to establish a collaborating group where most members are connected to the high-bandwidth network links while a minority has rather limited connection. If the network is capable to process—compress, down-sample, etc.—the data at the appropriate nodes (where the high and low throughput lines meet), the communication quality should not be reduced to the lowest common throughput denominator. The network must be capable of supporting complex communication patterns and must be able to process data internally. Robustness and failure resilience is another area, where more support at the network level is expected. While classical transport protocols like TCP support reliable data transmission, they are not appropriate for synchronous multimedia environment, where delays are unacceptable. It may be undesirable to wait for a timeout and then ask for a datagram retransmission, the network and applications themselves must be able to detect and immediately mitigate any data corruption or loss. Up to now, new requirements were served by different infrastructures tailored for a specific purpose. Nowadays, we need to merge them together in a network that uses packet transmission as its basis protocol—to do this successfully, new models, approaches, and techniques are necessary.

Computer networks are usually modeled as graphs—the nodes being network elements and attached computers and edges representing individual links. The model can easily be extended to multigraphs, which allow multiple line to connect any individual nodes. Although most computer networks are bi-directional, working a semi-duplex or full duplex regime, orientation can be added for explicit description of direction of flows (multiple edges used to represent the bi-directionality). As another step, we can add labels to the edges, representing some important properties like throughput or latency of each link. Labels on nodes can denote their properties, like different capabilities, latency of passing (bridging) data be-

tween edges of the node (the internal latency), size of internal buffers, etc. We can also speak about *internal* network, which is a part of the graph without any leaf node. It is also easy to identify end—i.e., leaf—elements.

Such a model is appropriate to study most usual flow patterns in contemporary computer networks, namely the send–receiver one. In this case, we have one node sending and exactly one node receiving a particular data flow. The basic network problem is finding a route between the communicating nodes, additional constraint is to guarantee available bandwidth and eventually other properties like overall latency or jitter. The route is usually the one composed from the smallest number of edges—the so called *shortest* path—but in some case any path could fit—this is the case, e.g., in the interdomain routing. The mechanism for creating a route can work on a flow basis—we speak about *connection oriented* networks—or on a datagram basis—the case of IP network. In the later case stability of the route is becoming additional important parameter that could influence the behavior of the whole flow (e.g., there is no reordering of datagrams within a flow in the connection oriented networks). Each path has one sending, one receiving, and zero or several *internal* nodes that are responsible for forwarding data.

However, as the networks were exposed to larger number of more sophisticated applications, more complex communicating patters emerged. The first one is a multicast, with still one sender but multitude of receivers. A simple extension is a communicating mesh, where every member of such a communicating group (the multicast group) could become a sender. Yet more complex communication patterns are seen in the *peer to peer* networks, where we may have multiple partially overlaid multicast groups communicating in parallel, it may use flooding, different cases of wave communication patterns, etc.

All the more complex communication patterns can still be expressed in our simple graph model using the sender–receiver paradigm. Multicast can be modeled by a set of sender to receiver$_i$ flows, but to express it correctly some kind of *coordination* (*synchronicity*) must be added to the model (data delivery to all receivers is expected to happen at the same time). Also, even in networks with unlimited bandwidth the simultaneous sending of all streams by just one element stress it above the optimal level (reducing efficiency of the communication scenario).

To deal with such complex communicating patterns more effectively, we have to extend our routing algorithm to find not paths, but whole subgraphs of the original graph. Flows going through such subgraph are more efficient than collection of individual send–receiver flows. The subgraphs represent *overlay networks*, that are specialized to transfer the particular flow pattern in the most efficient way.

When mapped back to the underlying network, the subgraphs extend the requirements on the internal path nodes. Simple forwarding (taking data from one link and sending them to another) is no longer sufficient, data must be duplicated and further processed to fit the communicating subgraph (overlay network) requirements. At the theoretical level this is just a simple extension, but propagating it back to the network proved to be very difficult, if not impossible work.

As an example, let's briefly discuss the IP multicast. It has been established as a family of protocols at the beginning of 80s in the last century. IP multicast is based on a family of multicast routing protocols (how to create the appropriate subgraph of the network) and its implementation requires support at each network element both for routing and also for multicast forwarding. The IP multicast includes nodes that do datagram duplication—they must be able to forward incoming data to two or even more output links. IP multicast does not

guarantee delivery of datagrams, does not provide any feedback to sender, it is in fact very simple extended forwarding scheme. All vendors of routers officially support multicast, yet it is not available on large parts of the Internet and the situation is not expected to change in the future. Although simple, multicast still can interfere with the basic sender–receiver communication patterns, imposes more load on routers (duplication is more complicated than simple forwarding) and the multicast routing protocols can introduce instability into the basic routing. As the result, multicast may not work properly or could be switched off by network administrators if they suspect it to be the cause of a problem they have with the network[1] [5, 8, 6, 7].

If the IP multicast situation is far from satisfactory, what we can anticipate with more complex extensions, where data have to be not only transmitted but also processed during transmission?

We must change the paradigm—instead of expecting the underlying network to provide all the advanced functionality and increasing complexity above sustainable levels, more *isolation* and *independent deployment* of support for complex communication (and data processing) patterns is the possible answer. The isolation is provided by the overlay networks, that take care of all the new functionality by themselves. The independence of deployment is achieved through the *user empowered* approach. The overlay networks are constructed and managed (often just temporarily) by their own users, without any need for specific support from network and its administrators.

Several years ago we started to build a network environment based on the user-empowered approach for transport and processing data in IP networks. We used the concept of active networks and designed and developed an *Active Element*—a programmable network node designed for synchronous data distribution and processing, configurable without administrator's right—and used it as the basic building block for construction of complex communication patterns.

The initial phase of our research was influenced by the network-centric view. We designed an active router [Appendix A], an extension of the classical router that allows users to define their own processing over individual data streams. The active network paradigm which introduced the active network elements, opened also the door to more user oriented approach. The active routers (and similar active network elements) are expected to be setup and operated by system administrators, with users "only" injecting smaller or larger programs to process their data within the network. Although the concept of active networks has been proved to provide the new functionality necessary to fulfill new requirements of data transmission and processing within the network, the whole idea collided with the conservative approach of network vendors and administrators. As the multicast experience demonstrated, it is very difficult to introduce new properties as they can interact in unpredictable way with the simpler, previously introduced protocols. Also, security concerns could not be overemphasized. A network programmable by end users is ripe for being taken completely by a hacker; this risk seen too high to be outweighted by the potential of new features.

[1] To further illustrate this problem, we have performed a quick survey of Internet2 Bigvideo group mailing list archive (`https://mail.internet2.edu/wws/arc/bigvideo/`). This list was in operation from May 2003 to May 2006. It focused on education and problem solving for users of high-end video technologies in advanced academical networks like the one operated by Internet2. The list was not limited to Internet2 community and there was a significant international contribution. As a majority of the advanced video tools use multicast, 212 of total 625 messages, i.e., 34% was spent on multicast testing and debugging.

At the same time as the active networks were developed, another paradigm that proved the value in giving control to end users[2] emerged—the peer to peer networks. They completely abandon the network-centric view, implementing in fact many already available network protocols once again, providing complete orthogonality (and independence) on the underlying network. The peer to peer networks are classical *overlay networks*, taking as granted only limited number of very simple properties of the underlying network and providing all the higher level functionality—searching, routing, etc.—by themselves.

However, the complete independence on the underlying network leads to inefficiency. The classical peer to peer networks could place their nodes only on the periphery of the network, where the users' stations are connected. The data distribution pattern required by the content (which the peer to peer network understood) may fit very poorly into the actual underlying network topology, overloading some lines while leaving other unused. Also, reliability of the peer to peer network is usually based on an overwhelming redundancy, when the same data are distributed, processed, and stored by many nodes—again a clear contradiction to the network-centric approach where the efficiency (the cost of the infrastructure) is one of the ruling paradigms.

We can see that the network-centric approach is highly efficient, but very slow in adopting new features and rather unfriendly to users. On the other hand, pure user-centric overlay approach, as represented by basic peer to peer networks is very inefficient (consuming more resources than needed in the optimal case), but it is able to introduce new features fast and can provide exactly the services the users are looking for. Another reason for that huge success is also their single purpose—the peer to peer networks are not trying to solve all the users' requirements, they focus on one service or just a small set of similar interconnected services.

Is it possible to take the positive from both approaches and leave out their negatives? Several years ago we decided to try this combination, moving from the network-centric to the user-centric approach, but not abandoning the network orientation completely. We extended the active router model to fit into the user-centric paradigm. The original active router and its implementation was based on Unix operating system and exploited both the kernel and user components. Its installation and deployment thus required system administrator's privileges that ordinary user may not have. As the next step, we completely redesigned the active router to become Active Element (AE), working in the user space of any operating system only. We obtained a fully user controlled element, that can be installed on any machine user has access to, without any specific privileges (e.g., on a server that is more strategically placed within the network than end user desktop machine). However, the AE design still followed basic network-centric pattern, being an evolutionary successor of active router, and thus became a keystone for the distribution and processing infrastructure, not a node in a peer to peer network. We still differentiate between an infrastructure and clients, but we put both into users' hands.

The user controlled Active Element is a very strong and flexible component to build different distribution schemes. We started with an infrastructure for virtual multicast. We used this infrastructure to study properties of the serial communication schema for group synchronous communication instead of the parallel communication model of the native multicast. While we had clearly demonstrated its advantages, especially in the area of security and reliability, the limited scalability remained the major disadvantage and it became our

[2]Should we be worried by the similarity to the *All power to people* paradigm?

natural next research target. Instead of using just a single AE to do all the processing and distribution, we designed a network of AEs with distinct control and data planes. This separation allowed us to use the peer to peer principles at the control plane, taking advantage of the properties of peer to peer networks like robustness and very high scalability. The inherent low efficiency of peer to peer networks does not play significant role, as the amount of control data is always limited. The result is an easily configurable and fault tolerant network of AEs with a reasonably high throughput capabilities.

However, the scalability is not one dimensional issue. While the network of AEs addressed the scalability in terms of number of clients supported, very high quality video (e.g., that used in the cinema theaters) generates so huge amount of data that may not be processed by a single AE. Therefore, we extended our work on scalability to increase the AE processing capacity through their internal parallelization. The parallelized AE runs on a cluster with fast internal interconnect and is capable of processing in near real time even 10 Gbps data stream.

All this research and development would not be complete without an actual deployment. Putting the AEs and their networks into production use provided a very valuable continuous feedback on their design while experimentally testing their properties. The AEs were used to build an infrastructure for collaborative environment used by several geographically distributed groups of researchers. Requirements from these groups initiated further research into support of advanced communication and collaboration features like moderating or subgrouping. The AEs started to play a role of directly controlled user tool to support these advanced properties. This confirmed the strength of the general concept of user empowered building blocks for data processing and distribution networks.

In another environment we used the idea of overlay network with AEs capable to provide new functionality for the stereoscopic video streams synchronization. A simple software implementation running on commodity hardware is able to synchronize two streams of stereoscopic digital video (DV, 25 Mbps) format successfully even when the original streams are highly de-synchronized. The penalty of the synchronization is increased latency, as the "faster" stream must wait for data in the slower stream, plus some processing latency is added to the final perceived delay. While this delay may be problematic in interactive implementation, we demonstrated that the AE-based synchronization element can be easily used for synchronized unidirectional stereoscopic streaming to multiple end users even in highly adverse and desynchronizing network conditions [Appendix F]. While the stereoscopic streaming may not be too common, this concept is usable for synchronization of stereo or multichannel (e.g., 5.1) audio streams or for synchronization of separately sent audio and video streams.

The real strength of the AEs and the whole concept of controllable overlay networks is demonstrated in the multi-point High Definition (HD) video distribution. If the HD video is to be used for a synchronous collaborative environment, uncompressed streams must be sent over the network. However, the required throughput of 1.5 Gbps per each stream was too high to be sent reliably over a native multicast in heterogeneous network over multiple administrative domains (even if it was available). The optimized AEs are able to replicate even such high demanding streams in near real time and were used to build infrastructure that supported one of the world first multipoint videoconferences using uncompressed HD video [Appendix H]. Later, improved AEs grouped into a network became key infrastructure for a virtual classroom that ran full semester and connected 6 sites on two continents

[Appendix M]. The Active Element network processed up to 18 Gbps bi-directional bandwidth, fully confirming the usability of the AE design.

In this thesis, we put together approaches published previously as separate papers, thus creating a new complex view of this field that generates new ideas and enables new applications. The simplest solution to user-empowered data distribution and processing is a central AE described briefly in Chapter 2. AE is a programmable modular active element, that can be run in the network easily without requiring any administrative privileges. The AE distributes and optionally also processes the incoming data, which allows for unique per-user processing capabilities—something that is impossible to do with traditional data distribution schemes like multicast. As any centralized solution, it has its advantages and shortcomings: while it is easy to setup and deploy, it has limited robustness and scalability, both with respect to number of streams and the bandwidth of a single stream. When more clients are collaborating or when higher robustness is needed, the AEs may be deployed as static or dynamic self-organizing AE networks shown in Chapter 3. This field has been studied thoroughly from the data distribution efficiency and robustness point of view by many groups previously [8] and the most relevant body of work is also referenced in Chapter 3. Our view here is, however, more general, focusing not only on mere multicast-like data distribution, but also on the possibilities enabled by additional data processing, operation in adverse networking environments, self-organization, etc. Another step forward needs to be taken when bandwidth of a single stream exceeds capacity of any single AE in the AE network. Utilizing properties of real-time multimedia applications and data distribution protocols, we have designed a distributed AE (described in Section 3.2) that can be deployed on tightly coupled clusters—but this solution becomes very complex when not only the data distribution but also data processing is required. We demonstrate applications which have been built on top of these technologies for synchronous data distribution and processing in Chapter 4. The thesis is concluded with a discussion of directions for future research in Chapter 5. The original work this thesis is based upon is collected in the Appendices A to M.

# Chapter 2

# Active Elements Evolution

Active Element (AE) plays a key role in our design. Its purpose is to process (multimedia) data and to forward them either to a next AE or to the target system. In classical networks, the router is an equivalent of an AE. To demonstrate the evolution of our understanding of AE role and architecture, let us start with a simple model of a classical router as shown in Figure 2.1[1].

A packet enters the router at one of its interfaces. It is stored in a shared buffer pool and all operations on it are performed "in place", supporting thus the zero-copy architecture. Packets are filtered and classified, and if they pass these steps their next-hop address is set. In the final stage, the packets enter the queue manager which puts them in an appropriate queue at the corresponding output port and through some interface the packets leave the router. The packet processing is controlled by the router management. In case of multicast router, the packets are also copied and put in several output queues, with appropriate next-hop addresses.

The responsibility for packet forwarding—in the sense of selecting the appropriate interface/port (queue)—lies with the routing protocols. Although the most important part in the classical router model, our model does not reveal details of the base network routing as this part is not directly influenced by the extensions leading to the model of an AE. AEs are used to develop overlay networks, which do not modify the base network routing but define new protocols on top of them.

The classical router does only very limited packet processing—usually only few items in the packet header (like TTL) and the next-hop address are changed, not the actual payload—and user has no direct influence on the router behavior. With the development of network services, new functionality was needed, leading to the development of the active network concept. Active networks were introduced to provide more flexibility—network with active routing and switching elements can be seen as a special type of distributed computing facility [23, 20], that enables completely new network usage paradigms. We started to be interested in active network principles as part of our work in the collaborative environments and their network support. The active networks promised to provide support for the real time transmission of the audio and video streams together with the possibility of processing

[1]The PC router was developed at MU, in collaboration with INRIA in early 90s. Although its architecture and performance has never been published, it became the most used network element in the early stages of the Masaryk University network backbone (and it was also widely used as part of the Brno Academic Computer Network till late nineties of the last century) due to its low cost and very high flexibility that allowed to introduce new routing protocols as they became available.

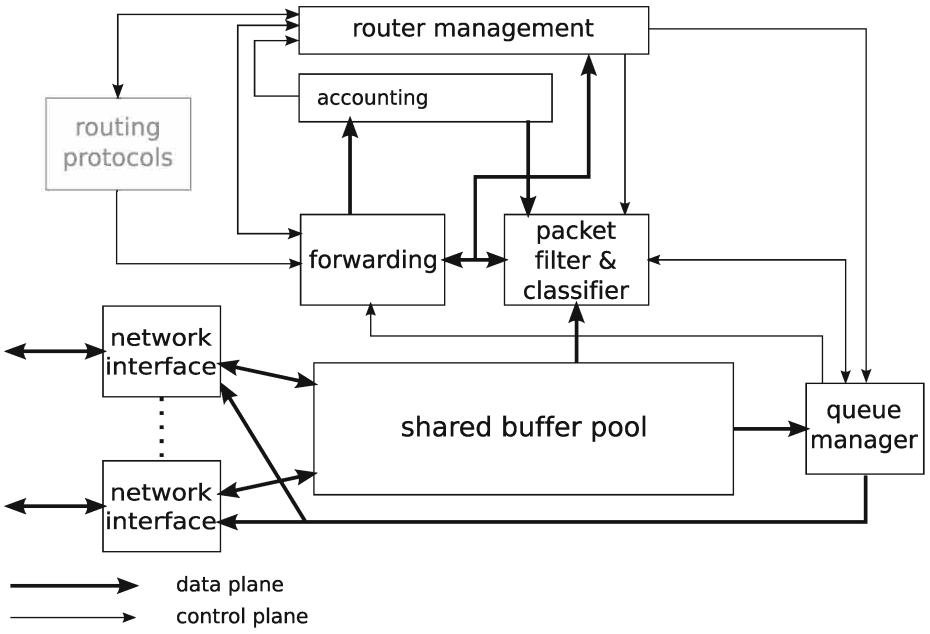



FIGURE 2.1: Model of general router

these streams directly within the network. Also, active networks dealt with the Quality of service (QoS) within IP networks, another requirement of the real time multimedia transmission.

The collaborative environments we developed needed rather sophisticated networks, combining complex QoS requirements, multicast distribution and content-specific processing. The best effort content-neutral approach of IP networks was insufficient and IP networks lack real QoS support—the available approaches like RSVP or DiffServ limited QoS parameters to raw bandwidth and transmission latency. There was no support for more structure- or content-oriented features like multi-priority packet drop and hierarchical data streams, for example.

As the first step, we extended the classical router model with active processing of data streams. Results of this work were published in [Appendix A] and the corresponding enhanced active router model is depicted in Figure 2.2.

The architecture of programmable (active) router kept all the classical router functionality and enhanced it with the possibility to process user programs. The user provided active programs can either be transmitted within each packet, or preloaded before the data packets belonging to a particular stream or fixed in the active router. We opted for the second option, with the program to be sent before the data. As a result, two phases can be distinguished:

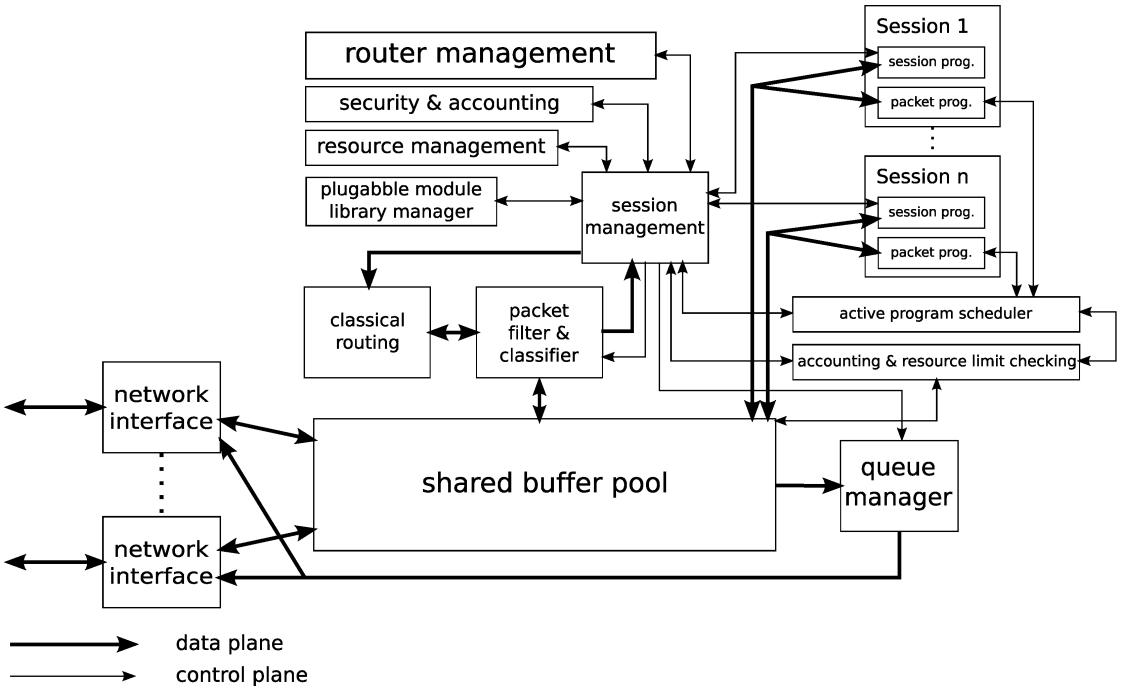



FIGURE 2.2: Model of active router

the first process controls the session establishment and management while the actual user data processing belongs to the second process. The first process has a role of a control plane, loading the user functions into all routers along the path between the data stream source and its destination. While the functions are usually preloaded before or during the actual connection setup, the model allows for on demand loading of additional user functions if a new requirement arises (this is taken care of by the active element control plane).

Comparing both Figures 2.1 and 2.2 we can clearly see new added components. Active packet processing is organized into sessions, with new router components added to manage the sessions inside a router.

Incoming packets are again stored in a shared buffer pool and classified. Non-active packets (i.e., packets that do not belong to any data stream recognized by the control plane nor recognized as control plane packets themselves) are processed in exactly the same way as in classical routers. However, if an active packet is recognized, it is processed according to its type—control plane or data stream. The control plane packets are responsible for connection and active program control, the latter are processed by the active programs.

When a control plane packet arrives, its content is processed by the session management module. This module is responsible for security checks (using information from the security and accounting module) and also checks whether the active router does have sufficient resources to run the active program and to process the user data. If the checks are affirmative,

the connection request is accepted, the next-hop active router is notified and the control establishment packets are sent to the next hop. When all the routers between the source and destination confirm the session establishment, the user data can be sent. These packets—the so called *active packets*—are marked (with a label defined during the path establishment process) and as such recognized by the packet classifier of each active router on the path. The active packets are processed by corresponding packets programs, loaded as part of the path (and session) initialization. The processing consumes some resources (CPU, memory, ...) and the resource utilization is continuously monitored by the appropriate module.

The active elements keep state information, which may lead to an overload situation, when not enough resources are available to fulfill new path establishment request. In such a situation, the new session establishment request may be refused, or currently running session programs may be asked to release some of already allocated resources. If the session establishment is refused, the previous active element (the last one that accepted the session establishment) can contact a different new active element to search for an alternative path.

The active network in fact creates an *overlay network* over the underlying conventional network that is expected to use "dumb" but fast elements for data transmission without any unnecessary features. In such a network, active elements are placed in key locations only, where special functionality is needed (typically near bottleneck lines, at the ingress and egress of the fast network core, on gateways between different networking technologies, or at important flow fork points). The active router model extends the "dumb" router functionality with a family of new protocols. These protocols relate to extensions of the active packet processing—network connectivity management and program initialization and processing.

Five years later we returned to the active router model when we started to work on enhanced QoS treatment. The classical QoS approaches deal with certain parameters like priority and queuing strategy on individual data flows. However, active routers introduce new complexity level. As user programs run on the shared active router, new parameters like processor time, amount of available memory, processor scheduling strategy are becoming relevant and they must be guaranteed, too. We proposed a QoS-enabled VM-based active router architecture that supports extended set of QoS related parameters [Appendix I] and provides strict isolation of user processes.

The generic modular architecture of active router, developed earlier, provided a very good basis for extensions to support complex QoS treatment. In addition, we slightly modified the original scheme to introduce use of virtual machines (VM) inside the QoS-enabled active router. With the virtual machines, user are able to upload not only their active programs into existing operating system environment, but they can upload whole virtual machine with its operating system. This way, the user's data are processed by programs running within their own environment, that includes also a specific operating system and its libraries. The virtual machine based architecture ensures strict separation of individual environments and user programs processing the data, but also allows for efficient scheduling of resources. The resources—CPU, memory, storage subsystem space and access, etc.—are allocated to virtual machines and controlled at this granularity level.

The architecture of our VM-ready active router is shown in Figure 2.3. The virtual machine architecture works with a Hypervisor (virtual machine monitor, VMM) which controls directly the hardware and provides the basic virtualized environment for the user's virtual machines. In Xen [2], that is the virtualization environment we use, the VMM is driven by `Dom0`. `Dom0` is the "overseeing" virtual machine which is responsible for creation and

destroying all user virtual machines—DomUs. In this environment, we had once again to decide which functionality will be shared and which is to be run inside a user virtual machine environment. In our design, the VMM (Dom0) keeps the shared buffer pool and is responsible for the low level interaction with hardware (e.g., interaction with network interfaces, the queue manager, and the packet scheduler) and is also responsible for the packet filtering and classification. The Dom0 is also responsible for the session management, including the resource management, security and accounting (more details on the resource management module and the virtual machine/active program scheduler module are given in [Appendix I]). This design keeps the most demanding packet processing—the filtering and classification—outside the overhead of virtualized network interfaces, that still have much lower performance than the physical interfaces themselves[2].

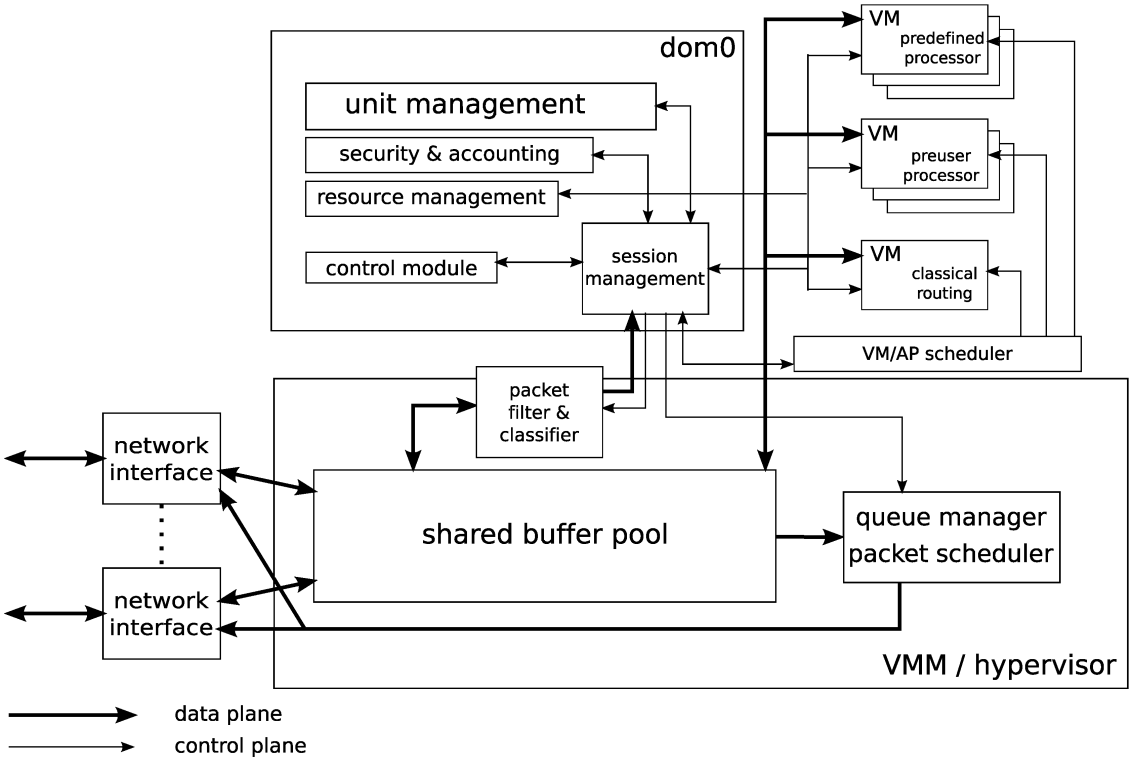



FIGURE 2.3: VM-ready active router architecture

The virtual machines managed by the session management module could be either fixed, providing functionality given by system administrator, or they can be user-loadable. A typical example of the fixed virtual machine is a virtual machine providing classical routing as shown in Figure 2.3. It is also the example of an optional module, as the AR can run with-

[2]While the performance of virtualized network interfaces is improving, with many groups working on new highly efficient design, the proposed architecture is still valid, as the low level packet processing is shared by all virtual machines.

out the classical routing if only "active" traffic passes through the AR, i.e., if it works in a dedicated overlay network. Users can upload a full virtual machine—including operation system and specific libraries—or they can use provided virtual machines to upload only an active program.

The VM-extended router provides a predictable and guaranteed access for active sessions to system resources. The shared part—the packet classification, filtering, and the management of interfaces and the buffer pool—has rather well understood resource requirements. Also, the resources needed to manage the whole active router and its virtual machines and to take care of security (implemented in `Dom0`) can be kept under strict control. The least predictable part— active programs provided by users—is encapsulated in virtual machines and the resource management is able to control distribution of CPU and memory to them in rather straightforward way. Users are not limited in the complexity of their active programs—they can upload full user-defined virtual machines—while the system keeps these programs well encapsulated. This architecture provides the ability for services to evolve seamlessly without changing the underlying model.

In parallel to the development of the VM-extended active router model (and in fact a little bit earlier) we started to accentuate the user empowered paradigm as the best way to give users the flexibility they are looking for without compromising the simplicity of the basic underlying network. The active router model has been modified to become Active Element (AE), which began as a laboratory experiment and evolved to an important component of a production collaborative environments.

Consistent application of the user empowered paradigm on active router architecture demanded crucial change in the architecture—leaving the kernel space. As a consequence of this major step, other changes had to be included in the final AE design.

AE [Appendix B] is a programmable element designed for synchronous data distribution and processing while minimizing the latency of the distribution. The word "reflector" is also being used in this context, while it only refers to data distribution capabilities. Since our approach is far more general and close to the idea of active networks, we have decided to use the *Active Element* name. The architecture of the AE is flexible enough to support implementation of different features while leaving space for easy extensions. It runs entirely in the user-space and thus it works without requiring administrative privileges on the host computer, adhering to the user-empowered principle.

The architecture of an Active Element is depicted in the Figure 2.4. The main difference from the active router architecture is the consistent transfer of all parts to the user space. Network interfaces are changed to become the network listeners. Network interface is a hardware part of the router and we usually expect two or more interfaces on any one router. Network listener of an AE is a process and there may be many more network listeners than there are physical network interfaces. Shared buffer pool was replaced by a more general model of shared memory. Input for AE are not only listeners but also Message interface modules—special types of listeners for control communications [4]. As in the active router model, we have a session management module with the same function in the AE. Packet programs and session programs are transformed into packet processors and session management, but at the higher level of abstraction, the functionality remains the same. The module for accounting and resource limit checking is missing in AE. This is due to the fact that AR is shared by many users in the same time, but the AE is expected to be used by just one user (or few cooperating users) which can take care of eventual congestion by himself—

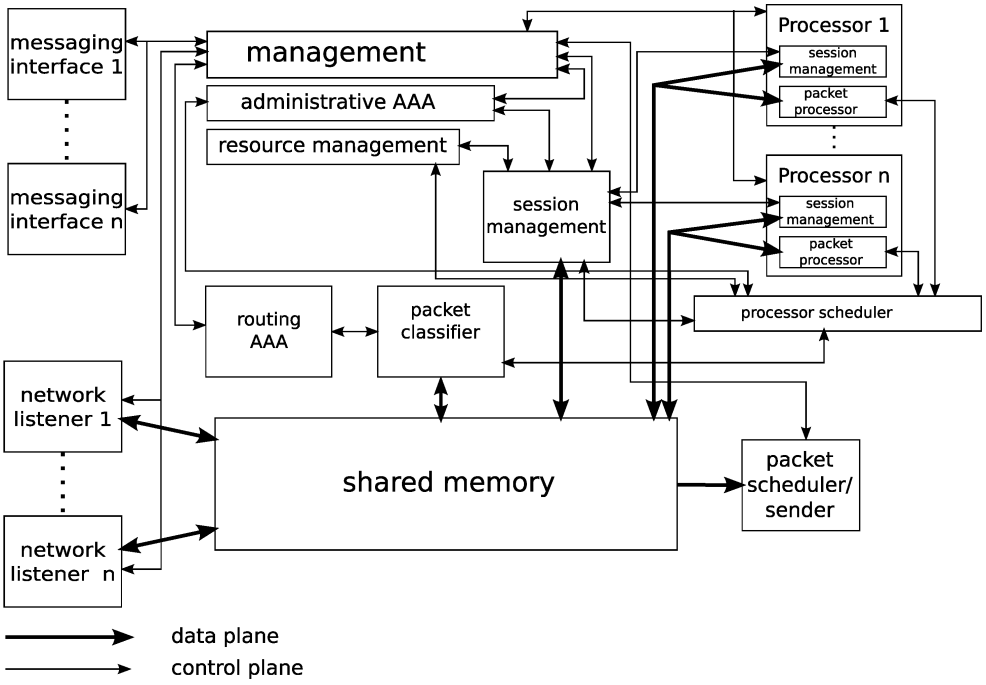



FIGURE 2.4: Active Element architecture

this is one of the advantages of user empowered approach, as some complex processing (like the resource accounting and limiting) are no more needed.

In this chapter the major steps of the Active Element development were described. With the AE, we have a modular architecture fulfilling requirements for processing and replication of data. While being a part of the infrastructure, the AE is fully end user controlled and provides a means to process data directly within the network. Still, it is a centralized solution which limits the number of clients, bandwidth of data going through it and the available processing power. In the next chapter, we will describe how these shortcomings are overcome, while keeping the full power of the AE in the hands of end users.

# Chapter 3

# Scalability

Despite the value of the AE, its usefulness is limited by scalability constraints when deployed as single centralized element. In this chapter, we describe our approach to the following two scalability aspects: (a) scalability with respect to number of clients and streams and (b) scalability with respect to processing requirements of an individual streams. Both of these can be handled by distributing the AE in certain ways. The first problem occurs when the AE is to serve a large number of clients or streams—we have solved this problem using *AE networks* described in Section 3.1. The second aspect becomes crucial when processing requirements of a data stream exceed processing capacity of any single instance of AE and thus the network of AEs is not a viable solution (or at least not *per se*). In order to overcome this limitation, the AE architecture has been extended to work in tightly coupled distributed environment by parallelizing the processing of an individual stream. This form of AE has been called *distributed AE* and is further described in Section 3.2.

There is also an additional aspect associated with either way of distributing the AE: robustness with respect to failure of AE nodes. This can be automatically handled by the concept of AE networks, that can re-route the traffic and redistribute the clients among the AEs as discussed in [Appendix E]. The problem is also mitigated by taking advantage of the user-empowerment: when the community of communicating users notices robustness-related problems, new AEs can be easily started by the members of the user community as needed.

## 3.1 AE Networks

The limitations of the communication model based on a single AE are clearly visible. The central communication model does not scale well and the number of potential clients is rather limited, with the actual numbers depending on amount of data being transmitted per stream. We lifted this limitation by transforming the concept of the single user-empowered AE into a user-empowered network of AEs in [Appendix C], [Appendix E], [Appendix G]. Individual AEs are interconnected by tunnels, thus creating an overlay network that balances its load to achieve high scalability.

When designing the concept of the AE network, we decided to use the out-of-band approach to separate transmission of control information through the *control plane* from the actual data transmission using *data plane*. This allowed us to optimize both planes independently given their different goals. The control plane must provide maximum robustness,

even at the cost of lower performance—control messages, that are relatively seldom compared to the actual data being distributed, must be delivered as reliably as possible via the control plane. On the other side, the data plane has to be optimized for maximum transmission performance and minimal latency, leaving the control plane with the role of reacting to any communication problems that may occur.

The control plane is responsible for management of the AE network. It provides monitoring and failure detection, reconstruction of the network after a node or link failure. Putting these requirements together, we have opted for a peer-to-peer (P2P) architecture for the control plane as it exhibits very high resilience, keeps the network connected even in case of failures, and retains the user-empowered property.

For the data plane, we do not incorporate any fixed static model; instead we rely on the information provided by the control plane to dynamically build the data distribution pattern based on supplied model rules. The changes in the AE architecture to support the extended model of AE network can be seen in Fig. 3.1.

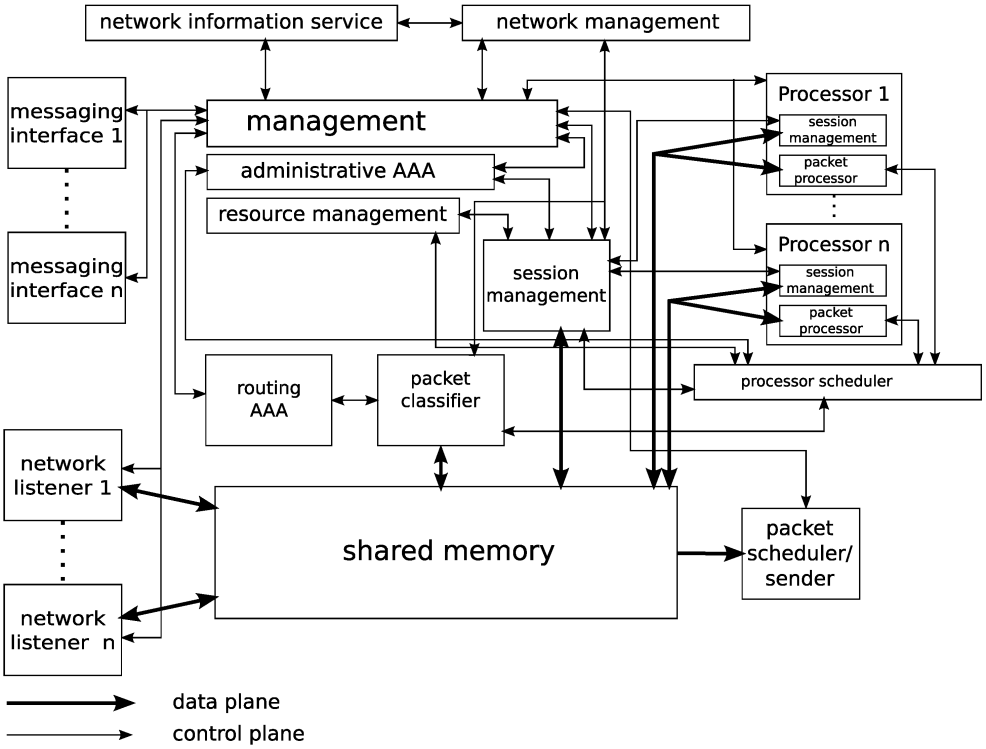



FIGURE 3.1: Architecture of Active Element with Network Management and Network Information Service modules.

The AE network support is implemented via two modules dynamically linked to the AE: the Network Management and the Network Information Service modules. The Network Information Service gathers and publishes information about the specific AE (e.g., available network and processing capacity), about the network of AEs, about properties important for synchronous multimedia distribution (e.g., pairwise one-way delay, RTT, estimated link

capacity), and also information on content groups (content and formats it is available in) distributed by the network. The Network Management takes care of building and managing the network of AEs, joining new content groups, leaving old ones, and reorganizing the network in case of link or node failure.

The data distribution plane is designed using loadable plug-ins to enable incorporating various distribution models. A number of suitable models has been proposed and studied previously by several independent groups in the past, most of which fall into one of the two categories: (1) mesh first distribution models, and (2) tree first models. These models can easily be incorporated into the AE implementation.

Prototype implementation of the AE networks with P2P control plane based on JXTA-C[1] has been demonstrated in [19] and [Appendix G]. A few simple optimizations to default JXTA settings improved the performance significantly for synchronous applications: with a limited number of nodes in the network (which is highly realistic in this case as size of truly interactive collaborative group is always limited) where down-time minimization is a key parameter, more aggressive communication and monitoring are acceptable despite increasing communication overhead.

The AE network is also designed to facilitate communication in adverse networking environments, i.e., environments where the network communication is obstructed by firewalls, network address translators (NATs) and various types of proxy servers. The data may be tunneled over TCP instead of usual UDP and it may even mimic HTTP traffic to tunnel the data over HTTP proxy. The AE may also be augmented by employing a VPN [Appendix K] such as OpenVPN[2], which boosts pervasivity, as it allows even tunneling through HTTP and SOCKS proxy servers. VPN also enables deployment of strong authentication and very secure data encryption protocols. Similar approaches have also been described in [1]. The solution that integrates these features directly into AE modules [21, 3] has significant advantages despite having a more demanding implementation: it allows for dynamic failure recovery properties in case of AE node failure or network link failure, as the client may automatically migrate to another AE node that is still available in the AE network and that is reachable for the client.

**User-Empowered Aspects of AE Networks** When distributing the AE architecture, the level of user-empowerment may be affected. The AE network has been designed with this approach in mind and the individual users can run the AEs for the network without any limitations and without requiring administrative privileges. The AE network support for the adverse networking environments makes it useful even in environments where other real-time communication tools fail because of unwillingness of network administrators to change configuration of the network to enable them. Thus the AE networks are at least as user-empowered as the centralized AE solution.

## 3.2 Distributed Active Element

Another scalability issue regarding both single AE and AE networks is scalability with respect to the bandwidth and/or processing requirements of each individual data stream. In

[1] http://www.jxta.org/
[2] http://openvpn.net/

order to improve this, we have parallelized the architecture of the actual AE node and designed a distributed AE ([Appendix J] and [Appendix L]). It is intended to be run on tightly coupled distributed systems with low latency network interconnection (e.g., commodity PC clusters with low-latency interconnection of Infiniband or Myrinet type). The low latency communication infrastructure is used for the control plane, while the external interfaces are used for the data plane. Given current convergence of high-bandwidth and low-latency network interfaces (e.g., Myrinet 10 Gigabit Ethernet), these two interconnects may technically converge, but conceptually we need to think about these separately in a way similar to AE networks.

The distributed AE splits a single stream into multiple sub-streams, which are processed in parallel. The distributed AE architecture comprises thee major modules as shown in Fig. 3.2: (a) distribution unit that takes care of ingress data distribution, (b) parallel AE units that do the actual data processing, and (c) aggregation unit that aggregates the resulting data onto egress network line(s). The parallel AE units are extended versions of standalone AE that support information exchange among the parallel units (e.g., state synchronization for data processing). The aggregation unit is a new module incorporating distributed version of packet scheduler/sender while the distribution unit is a completely new component.

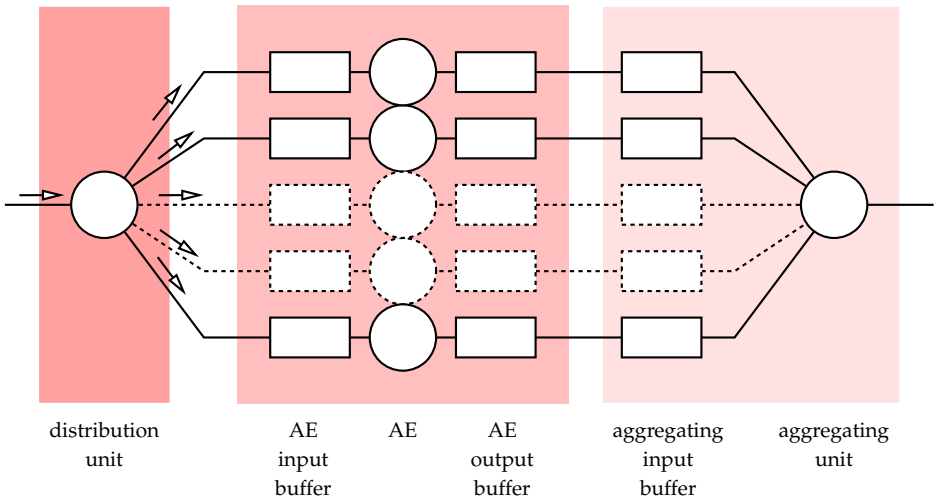



FIGURE 3.2: Model of the distributed AE.

Since data of a single stream are processed in parallel, there is a high probability of introducing packet reordering. The basic idea behind distributed AE utilizes the fact, that most of the synchronous multimedia applications use non-guaranteed data transport like UDP and thus they need to adapt to some packet reordering. However, significant data reordering may either not be adapted upon or it may result in latency increase as substantial buffering is necessary to order the packets back. Therefore we have introduced the Fast Circulating Token (FCT) protocol which limits packet reordering. Theoretical study based on buffer sizes in the underlying infrastructure has shown upper bound on the reordering that is significantly better compared to best-effort egress aggregation without any synchronization protocol. The receiving application can adapt its buffer size to this upper bound. On

custom hardware, the FCT protocol can be modified to provide no packet reordering at all (called Exact Order Sending in [Appendix L]).

The distributed AE with the FCT protocol has been prototyped using MPI and Myrinet low-latency interconnection. The experimental evaluation has been done using 10GE sender and receiver, while parallel nodes used GE interfaces for data plane and Myrinet-2000 network was used for the low-latency control plane communication. Packet distribution was implemented as user-space UDP library and the aggregation was performed by a commodity switch. The measurements revealed the distributed AE comprising of just 6 parallel nodes is capable of completely saturating sender machine (because the used 10GE card had its experimental performance limits at about 5 Gbps). The measurement series used up to 8 parallel units and it turns out that the experimental results with the FCT protocol are significantly better than the theoretical worst-case estimates and an order of magnitude better than the same distributed AE without FCT (more details can be found in [Appendix L]).

This approach is very different from general purpose load distribution systems like LACP IEEE 802.3ad protocol, which have to avoid the packet reordering and therefore a single data stream is processed sequentially[3].

The distributed AE can also be incorporated into an AE network using the same approach described in the previous chapter. However, since it runs on more elaborate infrastructure, the setup is more complex than for a single AE and thus the system has worse fail-over behavior compared to the network of simple AEs. Another complication of the distributed AE is in the processing of the passing data, which requires development of parallel programming paradigms similar to MIT StreamIt [24]. The processing may follow one of two possible approaches: (1) a context is maintained within one parallel unit only (requires either that (1a) all the data that need the same context are only processed within one parallel unit, or (1b) per-packet processing without a context), or (2) the context is maintained within a subset of parallel nodes using the low-latency interconnection of the cluster. These approaches establish a whole new field ideal for future research.

**User-Empowered Aspects of Distributed AE** As the tightly coupled distributed environments are usually hard to set up in a user empowered way, the distributed AE is less user-empowered solution. However, processing capacity of individual AEs raises along with increase of processing capabilities of commodity hardware systems. This effect can be even observed in history of our publications, where AEs running on year 2004 hardware were capable of producing up to Gigabit of traffic [13], while the year 2005 hardware enabled producing up to 5 Gigabits as shown in [Appendix H]. AEs running on current generation hardware go even beyond that, getting close to fully saturating 10 Gigabit Ethernet. Thus user-empowered AEs and their networks are sufficient for vast majority of purposes and the distributed AEs are limited to advanced scenarios (e.g., distributing uncompressed 4K video at 6 Gbps per stream) where users are also administrators of their systems anyway.

[3]This is done by using data flow identifiers hash to assign each data flow to a specific link of the of the aggregated link group. Thus each single data flow must not exceed capacity of the single link.

# Chapter 4

# Applications

The evolution of an Active Element and a scalable solution for data replication and processing was described in the last section. All steps of the evolution were closely connected and inspired by practical applications. In this chapter, we will present some interesting applications, which we will use also to demonstrate the basic performance features of AEs. As the AEs evolved to suit various applications and they run on various architectures, the measured performance cannot be directly compared. Therefore, the performance measurements presented in this sections should be taken as a guide for understanding of the concepts and limitations of AE performance, not as an exhaustive enumeration of all performance aspects of the different AEs.

## 4.1 Videoconferencing support [13]

Videoconferencing support was historically the first application of AEs. Instead of using native multicast, we used AEs to distribute data streams. We combined them with the MBone Tools [26] as client applications. The work is in detail described in my PhD thesis [13]. The main result was a demonstration that a native multicast (i.e., parallel) communication schema can be replaced by a serial communication (a star with AE in its center) without any negative effect on the user's perception of the quality of communication. Also, the videoconferencing support initiated further evolution of AEs and their features. We started with a simple UDP packet reflector [12] and gradually we incorporated management capabilities (e.g., user authentication and authorization) and extended the packet reflector into a full user controlled Active Element. The tests with various types of AEs had shown that the conveniently designed modular architecture of the AE has the same or at least very similar performance and quality of service parameters like delay, jitter, and packet loss as the original packet reflector and, even more importantly, as the multicast communication scheme. These experiments confirmed the usability of AEs for synchronous communication. The whole system for multimedia communication based on AE data distribution is described in [13].

## 4.2 Advanced Videoconferencing Support [Appendix D]

Later on, we demonstrated that Active Element is flexible enough to be used as a basic communication element for complex communication patterns. For several years we studied people's behavior during collaborative meetings and videoconferences. One specific feature which is missing in videoconferencing systems is support for creating smaller communicating groups, i.e., to allow users to speak with a subset of actual members of the whole communication group. In larger groups, the moderation of discussion is also necessary. Users with low to medium capacity connection often applied for optimization of data transfers. Another frequently requested feature was support for actual videoconference recording and playback.

In groups of eight or more participants, moderating is an essential capability for efficient communication [9]. We introduced the moderation feature as a special processing module within the AE. According to moderator's request the module decides which RTP and RTCP packets may pass through the AE from a particular person and which packets are discarded. Headers of RTP and RTCP packets sent by a person who was given the "floor" are altered in such a way that the packets from people chosen by a moderator to talk sequentially, form a single data stream regardless of the particular speaker [Appendix D].

When people meet to discuss some important issue, they often need to discuss it within a small subgroup in parallel with the discussion in the whole group (e.g., to communicate a common strategy). In such a situation, it is important to keep others from eavesdropping such a private discussion. In real world, the subgroup either moves away or uses whispering. Both of these solutions have certain disadvantages—people which are physically close to the whispering group are still able to overhear at least some parts of the discussion while when moved away, the subgroup members can not continue to participate in the primary discussion. Moreover, the members of the subgroup can be identified. To be integrated into a videoconferencing system, this functionality must be supported directly by the communication layer. Otherwise, the only way of satisfying the need for creating isolated subgroup is to create a stand-alone group, but this is the equivalent of moving away. Moreover, participants who want to create such a subgroup have to be physically isolated. By adding support for subgrouping into a communication layer, all disadvantages of non-distributed collaboration environment may be bypassed by separating a subgroup from the rest of a whole group by a "semi-permeable wall." Thus, private communication among members of a subgroup cannot be overheard by other people while a discussion outside the subgroup is still distributed to all (including subgroup members) participants. Naturally, this solution expects each participant to be isolated from every other one (e.g., heaving its own headset). Implementation of subgrouping within our AE-based communication architecture was quite straightforward as it does not require any new data processing facility. The module designed to support subgrouping just alters a list of clients each packet is sent to. A decision which clients should be removed from distribution list is based on a particular user's—the subgroup moderator—request specifying all members of a subgroup. Current design of the module uses IP addresses as users' identifier but it can be easily extended to support more advanced authorization approach (e.g., using the same mechanism as the moderating module for distributed AE architecture). More details on subgrouping support can be found in [Appendix D].

## 4.3 Video streams composition [14]

Video stream composition was the first application of the data processing capabilities of the AE and it was a direct reaction to user's need. In a large-scale group collaboration, too many windows may need to be supported at client sites (this is e.g., a typical scenario for AccessGrid[1]). Clients may not have sufficient processing power or desktop space to render them all. In such cases, it may be advantageous to down-sample the video streams and compose several of them into a single stream directly on the AE. The same technique is implemented in MCUs for H.323/SIP, but it was unavailable for MBone Tools. The first version of video compositor [14] has been adapted to fit into the modular AE architecture as a processor. This processor is based on the VIC tool [18] and thus it supports exactly the same set of video formats. The result of stream composition is shown in Fig. 4.1. Up to four video streams can be composed into one output stream. Input video formats are auto-detected, the processor is able to work with different layouts simultaneously. The output video format is configurable by the end user.


FIGURE 4.1: Example of video stream composition at AE using VIC video clients.

## 4.4 Stereoscopic video [Appendix F]

Another challenging application was a support for stereoscopic video streams. The main requirement is synchronization of the two video data streams. We interpret synchronization

[1] http://www.accessgrid.org/

as a special type of data processing and implemented it as such in the AE. We have created an Active Element with a module for stream synchronization [Appendix F].

The stereoscopic video stream is composed of two streams, each providing data for one eye. These two streams are either fully computer generated or the can be captured by two cameras in appropriate position to each other. Independent of whether the video streams are sent over the network separately or as a single merged network stream, the video streams have to be synchronized. The synchronization must be very precise, otherwise the stereoscopic perception of non–static scene is distorted (and watching it may have very bad physiological consequences). If we distribute the stereoscopic video to several sites, the synchronization is best done directly within the network. As the synchronization is just a special case of data processing, we used a synchronizing AE module that keeps multiple RTP packet streams synchronized, providing timely and in-order delivery to connected clients.

RTP packets include relative time-stamping information which may differ both in time base and time increment for different source streams coming even from several applications running on a simple client computer. Conversion between relative time and absolute time is performed using information sent in RTCP packet that are sent with lower frequency for each stream. RTCP packets contain both relative time-stamp and absolute time-stamp in NTP format. Therefore, after receiving two RTCP packets it is possible to calculate both relative time base and increment. If source machines have their clocks synchronized e.g., using NTP protocol it is also possible to synchronize streams coming from different machines. Since UDP packets provide no guarantees in terms of delivery, it is necessary to perform two steps when synchronizing: reorder or discard out-of-order packets and match time information for packets in different streams to send packets to clients synchronously. It is necessary to understand that due to the packet processing, the latency of transmission increases. This is not desirable for interactive applications like videoconferencing where even small latencies in order of a few hundreds of milliseconds might induce communication problems (e.g., when one person tries to interrupt the other to express his/her opinion).

The experimental evaluation of this approach is also published in [Appendix F]. The AE internal delay in synchronization mode was around 68 ms and if the streams are not desynchronized for more than 20 ms, this delay is not increased. However, if the inter-stream delay increases, it has a very negative effect on the total latency, with the worst case occurring when the inter-stream delay is around 100–200 ms. When increased above 200 ms, the absolute and especially the relative penalty decreases, as the AE has enough time to process both streams practically independently. For very high inter-stream delay the final penalty is around 10 % of the inter-stream delay, which is probably acceptable overhead for fully software-based synchronization. All the output streams are fully synchronized and the results demonstrate that the synchronization is practically absolute.

## 4.5 Stream transcoding [16]

Stream transcoding is a typical application of processing power of AE. For live video stream distribution from several lecturing halls at the Masaryk University, a transcoding processor for the video and audio streams has been implemented as an AE module [16]. It uses Video–LAN Client[2] (VLC) as the actual transcoding back-end, thus giving us a large variety of supported formats for both input and output. The transcoding module communicates with

[2] `http://www.videolan.org`

the VLC in three ways: the source data is delivered using Unix standard I/O, the transcoded data is received from VLC using a local UDP socket in order to receive the data appropriately packetized, while a local `telnet` interface is used for control of the VLC.

For the specific application of live lecture distribution, two types of video stream sources are used: an MPEG-2 hardware encoder such as Teracue ENC-100 or a regular MPEG-4 streaming PC with video capture card and VLC installed. In both cases the video streams are generated as a standard MPEG Transport Streams (MPEG-TS) at 2 Mbps and sent via unicast to the AE for further processing. The original data is also available to the students either using unicast or multicast from the AE, or they can watch transcoded video from the gateway AE. Students then use VLC again for rendering the streams at their computers. This allows to provide students connected to high-speed networks with the maximum quality video, while students with slower networks (e.g., in dormitories) are also supported and may participate in the class using transcoded streams with a lower bitrate. Depending on the settings, the transcoding can consume a considerable amount of processing power and therefore the transcoding AE has significantly lower distribution capacity. As a result it is set up at the beginning of the low bandwidth link working as a gateway or bridge only, while another AE is used to actually distribute the transcoded data at large.

In order to evaluate efficiency and scalability of this solution, a series of performance and latency measurements with variety of transcoding settings was performed. The results show that conservatively set transcoding AE scales well for the number of connected clients. Moreover, the transcoding imposes a constant load on the AE. Under the circumstances described in [16], up to 700 clients was saturated with the transcoded stream. The latency introduced to the transcoded stream is caused by the VideoLAN client used for the transcoding. When compared with it, the latency introduced by the AE itself is negligible. The conclusion taken from the latency measurements is that there is a considerable amount of buffering done in the VideoLAN client and thus the transcoding AE based on the VLC module is suitable for unidirectional video distribution but not for interactive videoconferencing.

## 4.6 Collaboration in Adverse Networking Environments and Secure Collaboration [Appendix K]

The real-time communication for healthcare purposes is unique because of two classes of related requirements: (i) security and (ii) ability to operate even in heavily protected networking environments. The security is necessary as specialists often need to discuss very sensitive patients' data. Because of the security requirements, the healthcare institutions are usually trying to implement the most restrictive networking environments. It is not uncommon to find internal hospital network shielded by a firewall, hidden behind a NAT, opened only for HTTP traffic, which has to pass through two tiers of proxies. However, even the specialists from this hospital need to communicate with their colleagues. The AE approach combined with VPNs has been successfully deployed for several healthcare related projects [Appendix K] and we were able to include even the hospitals with the most restricted access mentioned above.

In order to evaluate influence of incorporation of OpenVPN into the collaborative platform, we have measured a number of parameters critical for real-time multimedia communication using different VPN modes. The measurement testbed comprised one client and one VPN server, interconnected with high-speed backbone network link with capacity

above 1 Gbps spanning about 250 km. The results of measurements are in detail described in [Appendix K]. To conclude, UDP based VPN is very safe and has minimum impact on the traffic. CPU requirements slightly increase for the UDP based VPN compared to the TCP based VPN, due to the application-level packet loss recovery and congestion control, which is marginally less CPU efficient compared to the kernel-based TCP implementation. TCP-based VPNs also perform very well provided they are on a low-latency network with very low packet loss, so that congestion control algorithm doesn't influence the data flow significantly. However, TCP based VPN is inappropriate on networks with higher packet loss. If the HTTP proxy is of good performance, it has minimum impact on the performance, too.

When evaluating the performance of this solution subjectively, the media streams are fine and the overall quality is very good. Another important feature that was developed in this field is efficient aggregation of individual media streams—not only the video streams as discussed above—as some of the institutions, especially in developing countries, have only very limited Internet connection capacity. The aggregation, when combined with downsampling, can support multi-party communication even in such restricted environments (e.g., the total capacity of the link connecting academic network of Cyprus to Greece is only 10 Mbps, leaving very limited bandwidth for the videoconference communication).

## 4.7 HD Video Distribution [Appendix H], [Appendix M]

The AEs have been used routinely by different groups for collaboration, mostly with MBone Tools[3], DVTS[4], and uncompressed HD video based on UltraGrid [Appendix H]. A recent demonstration of uncompressed HD video with bandwidth usage of 1.5 Gbps per data stream at SuperComputing'06 conference[5] used a network with 3 optimized HD AEs in StarLight (Chicago, USA) and achieved sustained aggregated data rate of 18 Gbps without any packet loss. As an alternative setup, we have also used a combination of an AE with multiplication on optical layer (optical multicast), which is, however, far from user-empowered as it requires both direct access to the Layer 1 network and installation of specialized hardware directly into the network. The high-performance static AE network has also been used in production for uncompressed HD video distribution for a distributed class on high-performance computing taught by prof. Sterling at Louisiana State University [17]. In this case, dedicated $\lambda$-circuits spanning 5 institutions across the USA and one in the Czech Republic were used. A static configuration was the most appropriate as the circuit topology was statically configured. The 1.5 Gbps streams were distributed up to 5 locations and the schema of topology and other details are in [Appendix M].

## 4.8 Conclusions

This chapter covers several examples of the AE deployment to support experimental as well as production applications. The application expose different aspects of the AE properties and together clearly demonstrate the usability of the AE (both the concept and the individual implementations) as a general building element for network support of synchronous multimedia application.

[3] http://www-mice.cs.ucl.ac.uk/multimedia/software/

[4] http://www.sfc.wide.ad.jp/DVTS/

[5] https://sitola.fi.muni.cz/igrid/index.php/SuperComputing_2006

# Chapter 5

# Conclusions and Future Work

This thesis presents our approach to the problem of efficient synchronous data distribution and processing in contemporary networks. The proposed solution is built around the Active Element, a basic building block based on concepts of *overlay networks* and the *user-empowered approach*. The active element is a general framework, that could be tailored to support different environments, data stream patterns and configurations. The AE can be used as a single central entity, offering full end user programmability and control. If more robust, failure resilient, and scalable solution is needed, the AEs could be combined into a network, with a peer to peer control plane and dynamic data plane. A distributed AE, running on a dedicated cluster, provides enough computing power to process, in the real time, even the most demanding data streams. The single and distributed AEs, as well as their networks, are deployed and controlled directly by the end users, who can also provide programs for processing individual data streams. This user oriented deployment allows to build AEs and their networks in an ad hoc manner, as an overlay network best suited to particular end user's needs. Unlike in the peer to peer networks, the AEs are still part of the network *infrastructure*, but controlled by end users. The user controlled infrastructure approach is well suited to serve as a distribution and processing backbone of many applications that either need some level of (end user) control (e.g., corporate networks) or that push the network distribution capacity to the edge (and could not sustain the very high overhead and inherent inefficiency of the peer to peer networks).

To demonstrate the flexibility and potential of the AE framework, we used it to support several different applications, as also reflected in this thesis. The high flexibility of the AE has been demonstrated by many groups that use it as a distribution element for their routine videoconferences. The AE framework flexibility allowed to use AE for simple collaboration with low level audio and video quality, but also provided a distribution element for an uncompressed HD videoconferences with extreme bandwidth requirements. AEs were used to support scientific visualization combined with the real time steering of simulations. AEs succeeded in the data streaming, transcoding video on demand, as needed by the client's appliances.

As the best performance and efficiency is achieved when the AEs are placed in the strategic nodes of the underlying physical network, we have permanent AE servers at several GigaPoPs (Point of Presence) of the Czech academic backbone CESNET2 (in Prague, Pilsen, Liberec, and Brno). We also already succeeded in having AEs permanently installed in the most important exchange point of the global optical network—at StarLight in Chicago

(USA). We plan to install AEs in other locations, too, including the most important European exchange point in Amsterdam. This way, we are becoming able to provide a physical infrastructure for global experiments with the data distribution and processing capabilities of AEs. Using virtualization, this testbed will be available also for other interested parties.

Despite the already demonstrated practical utility of the AE framework, many theoretical and practical questions remain unanswered. In the future, we would like to continue our research in several areas, including further development of the Active element as a processing entity, extending the "intelligence" of the network of AEs, improving the data processing capacity of the distributed AE.

At a single AE level, we work on extension of the quality of service to the multi-level QoS approach that will allow strict separation both at the user and data levels. This will be based on the already introduced virtualized AE, providing more extensive support for programmability at the full virtual machine level. More work will be focused on better scheduling strategies, making sure the guaranteed data processing properties (speed, latency, jitter, etc.) are really met within a shared AE.

We need much more "intelligent" behavior of the network of AEs. Signaling protocols for better diagnostics must be developed, providing accurate information about failures and bottlenecks both for the AE network and its users. An open issue is a mapping of the overlay AE network onto the physical network and its nodes. Not only the initial distribution of AEs, but especially a reaction on failures and transient and permanent link and node break downs is a vital property of the AE network. We need to study languages for the static and dynamic description of the network topology and its state and to combine these with the AE network self-organizing properties. Also, the extent of the fully automatic self-organization (as currently developed for, e.g., Skype or VRVS successor EVO (Enabling Virtual Organizations [11]) compared to the user's power to organize the network directly must be studied, as the extremes (full user control and fully automatic control) are not optimal. This work could also benefit from the recent research at the area of network coding [10, 25].

Appropriate programming paradigms suitable for the distributed AE must be studied. A potentially interesting approach has been proposed in the MIT StreamIt system [24], that enables efficient parallelization of stream processing based on sent data structures and processing dependencies. Providing a suitable paradigm and a language to use it, end users will be able to describe the properties of transmitted data streams in a way that will allow (semi-)automatic parallelization at the distributed AE.

Further research in the security implications of the whole AE framework will be also necessary to better understand potential threats. We plan to exploit the user empowered paradigm to include users directly as part of the security threat mitigation (better diagnostics, new authentication and authorization schemes providing an intuitive environment, etc.).

And naturally, we will continue to support the use of AEs in different applications, as this deployment provides the most important feedback for further research.

# Bibliography


[1] L. Alchaal, V. Roca, and M. Habert: "Offering a multicast delivery service in a programmable secure IP VPN environment." In *Networked Group Communication, Fourth International COST264 Workshop, NGC 2002, Boston, MA, USA, October 23-25, 2002, Proceedings*, pages 5–10. ACM, 2002.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield: "Xen and the Art of Virtualization." In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-757-5.

[3] T. Bouček: *Kryptografické zabezpečení videokonferencí (Securing videoconferences using cryptography)*. Master's thesis, Military Academy of Brno, Czech Republic, 2002.

[4] J. Denemark, P. Holub, and E. Hladká: *RAP – reflector administration protocol*. Technical Report 9/2003, CESNET, 2003. `http://www.cesnet.cz/doc/techzpravy/`.

[5] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen: "Deployment issues for the IP multicast service and architecture," *IEEE Network*, 14(1):78–88, 2000.

[6] F. Dressler: "Availability analysis in large scale multicast networks." In *15th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS2003)*, volume I, pages 399–403, November 2003.

[7] F. Dressler: *Monitoring of Multicast Networks for Time-Synchronous Communication*. Ph.d. thesis, University of Erlangen-Nuremberg, May 2003. `http://www7.informatik.uni-erlangen.de/~dressler/publications/dissertation.pdf`.

[8] A. El-Sayed, V. Roca, and L. Mathy: "A survey of proposals for an alternative group communication service," *IEEE Network*, 17(1):46–51, January/February 2003.

[9] D. R. Forsyth: *Group Dynamics*. Wadsworth Publishing Company, 3rd edition, 1999.

[10] C. Fragouli, J.-Y. L. Boudec, and J. Widmer: "Network coding: an instant primer," *SIGCOMM Comput. Commun. Rev.*, 36(1):63–68, 2006, ISSN 0146-4833.

[11] P. Galvez: "From VRVS to EVO (Enabling Virtual Organizations)." In *TERENA Networking Conference 2006*, Catania, Italy, May 2006.

[12] J. Highfield: *UDP packet reflector hacks – RTP unicast mirror `rum`*. `http://spirit.lboro.ac.uk/mug/mug.html`.

[13] E. Hladká: *User Empovered Collaborative Environment: Active Network Support*. PhD thesis, Faculty of Informatics, Masaryk University Brno, Czech Republic, 2004.

[14] V. Holer: *Slučování videostreamů (Videostream Merging)*. Bachelor Thesis, Faculty of Informatics, Masaryk University Brno, Brno, Czech Republic, 2003.

[15] J. Jo, W. Hong, S. Lee, D. Kim, J. Kim, and O. Byeon: "Interactive 3D HD video transport for e-science collaboration over UCLP-enabled GLORIAD lightpath," *Future Generation Computer Systems*, 22(8):884–891, 2006, ISSN 0167-739X.

[16] M. Liška and J. Denemark: *Real-time transcoding and scalable data distribution for video streaming*. Technical Report 30/2006, CESNET, Praha, Czech Republic, 2006. `http://www.cesnet.cz/doc/techzpravy/2006/rum-streaming/`.

[17] L. Matyska, P. Holub, and E. Hladká: "Collaborative framework for distributed distance learning." In *Proceedings of the 4th High-End Visualization Workshop*, pages 40–45, Obergurgl, Tyrol, Austria, 2007. ISBN 978-3-86541-216-4.

[18] S. McCanne and V. Jacobson: "vic: A flexible framework for packet video." In *Proceedings of ACM Multimedia'95*, pages 511–512, San Francisco, CA, USA, November 1995.

[19] M. Procházka, P. Holub, and E. Hladká: "Active element network with p2p control plane." In *Proceedings IWSOS 2006*, volume 4124 of *Lecture Notes in Computer Science*, pages 257–257. Springer-Verlag Heidelberg, 2006, ISBN 3-540-37658-5.

[20] K. Psounis: "Active networks: Applications, security, safety and architectures," *IEEE Communication Surveys*, 1999.

[21] Z. Salvet: *Enhanced UDP packet reflector for unfriendly environments*. Technical Report 16/2001, CESNET, Praha, Czech Republic, 2001. `http://www.cesnet.cz/doc/techzpravy/2001/16/`.

[22] T. Shimizu, D. Shirai, H. Takahashi, T. Murooka, K. Obana, Y. Tonomura, T. Inoue, T. Yamaguchi, T. Fujii, N. Ohta, S. Ono, T. Aoyama, L. Herr, N. van Osdol, X. Wang, M. D. Brown, T. A. DeFanti, R. Feld, J. Balser, S. Morris, T. Henthorn, G. Dawe, P. Otto, and L. Smarr: "International real-time streaming of 4K digital cinema," *Future Generation Computer Systems*, 22(8):929–939, October 2006, ISSN 0167-739X.

[23] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, G. J. Minden, and D. J. Wetherall: "A survey of active network research," *IEEE Communications*, 35(1):80–86, January 1997.

[24] W. Thies, M. Karczmarek, and S. Amarasinghe: "Streamit: A language for streaming applications." In *Proceedings of the 11th International Conference on Compiler Construction*, volume 2304 of *Lecture Notes in Computer Science*, pages 179–196, Grenoble, France, 2002. Springer-Verlag Heidelberg, ISBN 3-540-43369-4.

[25] R. W. Yeung, S. y Li, and N. Cai: *Network Coding Theory (Foundations and Trends(R) in Communications and Information Theory)*. Now Publishers Inc., Hanover, MA, USA, 2006, ISBN 1933019247.

[26] *MBone tools*. `http://www-mice.cs.ucl.ac.uk/multimedia/software/`.

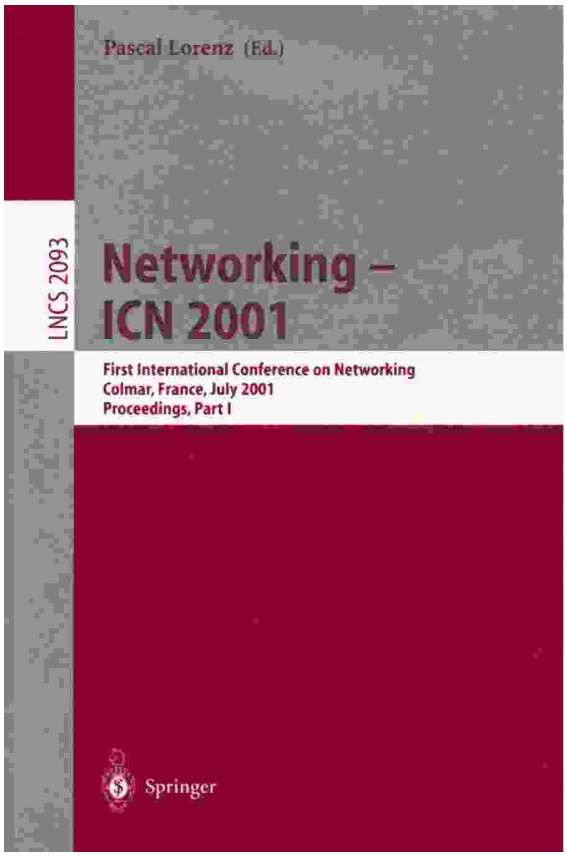
# Appendices

# Appendix A

# Active Network Architecture: Distributed computer or transport medium

by Eva Hladká and Zdeněk Salvet

In First International Conference on Networking, ICN 2001, Colmar, France, July 2001. Proceedings. Lecture Notes in Computer Science 2093, Springer-Verlag Berlin, 2001. pp. 612–620, 8 p. LNCS 2094. ISSN 0302-9743.



# An Active Network Architecture: Distributed computer or transport medium

Eva Hladká[1] and Zdeněk Salvet[2]

[1] Faculty of Informatics
eva@fi.muni.cz
[2] Institute of Computer Science,
Masaryk University, 602 00 Brno,
and also CESNET, z.s.p.o., Prague
Czech Republic
salvet@ics.muni.cz

**Abstract.** Future computer networks must be more flexible and faster then today. Active network paradigm is the way how to add flexibility to networks. During the last five years, several active network architecture models were presented. A new one, based on model of active nodes is presented here. The key features of this architecture are the separation of session or connection management functions from the bulk data packet processing functions and associated session management protocol that facilitates user control over active network processing. This architecture is designed to be sufficiently general to accommodate and build on top of any packet-based networking technology. The description of the model is followed by brief of prototype implementation using PC-class computers with NetBSD operating system.

## 1 Introduction

Introduction of new innovative Internet applications sometimes demands special network services which are very difficult or impossible to provide in efficient manner using contemporary standard networks behaving as a passive transport medium. Some of such new complex services may be naturally supported by adding various levels of user programmability to the routing elements inside the network (routers, switches, optical cross-connect systems, etc.). In the last few years, several of such programmable network architectures are being developed, commonly called "active networks".

Active networks were introduced in order to provide more flexibility within the network itself — network with active routing and switching elements can be seen as a special type of distributed computing facility [1], [2]. Like other types of distributed computing facilities (or even more so), this may pose a lot of problems to solve, depending on type of services supplied to (end-)user. These problems range from need of high-level authentication, authorization, and accounting schemes and policies usable with multiple administrative domains through task scheduling and resource allocation, reservation, and management

to lower-level performance, scalability, and device management issues. Also, in some cases confidentiality, non-repudiability or other special security services may be required. While not all these concerns are applicable to all active network architectures and service levels that may be supported, it is important to make sure they have been addressed before real-world deployment of active network.

Active networks provide excellent development environment for new network protocols. Our interest in active networks stems from this ability, combined with the interest in the real time transmissions of the audio and video streams and the associated requirements for quality of service (QoS). We find currently available technologies insufficient for use in situations where more sophisticated QoS management is needed. In typical IP networks, best-effort approach is used, and even best available implementations of RSVP and ATM protocols are limited to raw bandwidth and transmission delay management only. There is no support for more structure- or content-oriented features like, for example, multi-priority packet drop and hierarchical data streams. (Moreover, ATM is still and may remain too expensive for connecting ordinary end-user machines.) We try to construct AN architecture, which will help us not only to develop new network protocols tailored to new applications, but also gives us possibility to implement new QoS capabilities.

We have developed an active network architecture using the concept of active nodes and we are working on the implementation of this architecture using software routers based on the NetBSD (UNIX) open-source operating system. The main features of our active network model and its (prototype) implementation are influenced by the desire for support of novel QoS-oriented features that we plan to implement using the new active network model and its first implementation.

## 2 The New Active Network Model

The proposed active network architecture uses "active node" approach to active networking and concept of "sessions" similar to connections in connection-oriented networks or sessions in RSVP protocol.

Structure of an active node (router) plays a key role in our model. It is a network element which is able to accept user-supplied programs and to execute them. The processing of user code consists of two separate (communicating) processes. The first one controls the session establishment and management. It has the role of control plane in active router processing and includes (initial or later) load of user functions onto the routers along the path between source and destination address or addresses and execution of bookkeeping functions. The second part of user code, initiated by the first one, forms the central part of the data packet processing itself.

An *active session* consists of state information in active routers that controls handling of traffic among communicating endpoints. The most important part of an active session is the set of running instances of user code session programs. They manage other parts of state, i.e. spawning per-packet processing code, set-

ting input packet filters and classifiers and configuration of output queues, using the active node programming interface (API). The session itself can be controlled (created, spread to new active nodes, modified by setting various parameters, partially or fully destroyed) in response to user requests but also from the inside — by session programs. An active session can be managed either remotely using *session management protocol* or locally using management API of active node management software. In the typical scenario, user application communicates with first-hop active router using the session management protocol and starts the first instance of appropriate session program with desired parameters. The session program then computes some additional (next-hop) active nodes that should take part in active processing, starts new instances of session program on them and programs per-packet processing functions. The whole process repeats with any new active nodes used. If any failure occurs during session setup, crankback technique can be used to find alternative paths in network. The location of first-hop active router (set of routers) may be either included in the end user configuration data (manual configuration, DHCP data, etc.) or determined by lower-layer specific active router discovery protocol. The software running on the active node consists of four sets of programs:

- *session programs* that are supposed to take care of establishing connections to other active nodes, computing routing information, negotiation of QoS parameters and other functions related to user-defined session management. They are supplied by user or other active node during session establishment along with parameters and can be stopped or changed later during session lifetime.
- programs performing *per-packet processing* (forwarding, replication, policing, packet scheduling, etc.). These are supplied by user or other active node during session establishment as parameters or embedded parts of session programs. Session program can manipulate them and configure their relationships with standard packet processing modules (packet filters, queues, routing tables, output schedulers, device interface drivers).
- *basic system software* which implements session establishment and management protocol, basic execution environment for programs using native code, task management and basic resource management, security infrastructure (authentication and authorization policies), local session management API and facilities for device management by network administrator. This software can be changed only with administrative privileges.
- pluggable modules for optional *system software extensions*, e. g.
  - address/protocol families (the architecture is independent of lower networking layers)
  - authentication, authorization, and accounting protocols
  - program interpreters and runtime support (e. g. interpreters for interpreted languages, runtime linker, JIT compilers, module that downloads given URL for execution, etc.)

  The model is designed as general as possible in order not to be restricted by the architecture in later development and deployment stages. All extensible

information elements in session establishment and management protocol are designed with tagged polymorphic types. These modules can be downloaded by system software on demand from the code repository defined by the administrator or managed with device management facilities. IPv4 and IPv6 protocol families and C language binding will be supported by the prototype implementation.

The proposed active network architecture uses a connection-oriented (state keeping) approach. Obviously, this approach is more prone to scalability problems than capsule or "fixed function set" schemes but we believe that only small fraction of all data streams in typical network will require special QoS features or other algorithms running on active nodes, so the amount of state information which must be kept within the network elements will be almost always sufficiently low and not exceeding the processing and memory capacity of the active nodes. In overload situation, new session establishment request can be simply refused, currently running session programs can be asked to release some of already allocated resources, or alternative (possibly suboptimal in other situations) paths in active network can be searched. Second, our *active network* is meant as overlay network on more conventional network that may use "dumb" but fast elements for data transmission only (these are located especially in the core of the network) and the active elements will be placed only in key spots of the network where special features are needed (typically near bottleneck links, at the ingress and egress of the fast core, and on gateways between different networking technologies).

Using our session establishment and management protocol, it is possible to request change in parameters in atomic way (something that many signalling protocols cannot do without tearing down the running connection and trying to reestablishing it with new parameters) and negotiate permissible values of parameters (unnecessary trials and errors can be avoided in algorithms that are adaptable to various environmental conditions). The active nature of sessions have some other pleasant consequences:

- In case of failure or overload of some element (even not active element itself but some "dumb" router or physical link in between), connections may be re-established or rerouted automatically without any end user or network administrator intervention and without the need to implement these actions in performance-critical packet processing code.
- The session parameters can be changed dynamically by end users — such a change of parameters is processed in the similar way as the initial establishment of the session.

The active router model extends the "dumb" router functionality through a family of new protocols. The protocols can be divided to two groups relating to two phases of the active packet processing — protocols for network connectivity and program loading and those for processing loaded programs. Security of this active network model is based on authentication and authorization protocols not described in this paper and security properties of other used protocols.

One of our primary goals was to develop a general network protocol which would allow us to use existing network protocols or their modifications at the presentation and application layers. To achieve that, active node code may be authorized to process not only packets from active networking aware applications but also data sent by selected "legacy" applications. In such a case, separate application has to initiate the session with active node that resides on path used by application data and use appropriate input packet filter to capture the desired packet flow.

In session management protocol, the program is defined in generic way — only by code type, code length, and the program code itself. This allows different types of programs (e.g. compiled C or interpreted Java) to be run on the same active router. Due to split between session management and per-packet processing functions, it is possible to use different languages for these functions. Typical use of this feature would probably include running optimized native code in performance-critical per-packet program and implementing rest of code in convenient scripting language. Different performance characteristics of different types of code should be taken in account when calculating processing resource limits (CPU, memory, etc.), but this is an open issue in our architecture and subject to further research.

### 2.1 Data Plane

The actual processing of active data packets is done by program, which is (or whose identification is) supplied to the session program as part of its parameters. The session program is responsible for allocating necessary resources and setting parameters of per-packet tasks (either supplied by user or determined from the present active network state) and calling the active node API to start it up. The per-packet program is executed in a loop, this loop is permanent while the session is alive.

The active program registers active packets filter — e. g. the IP source and destination addresses, and the port number or precomputed flow label value — for this particular active connection and relies on the active router which sends all such packets to the active program loop. The packets are processed (e.g. modified, deleted, duplicated, ...) and than sent to appropriate active router port or ports (e.g. in case of multicast-like data). The fairness between different active programs running on the same active router is guaranteed by the active router core, which uses on one side the authorization data and on the other the statistics which are collected for each individual active program (CPU, memory consumption, etc.). The active programs are scheduled with respect to already consumed resources and to the actual resources they are authorized to use. In such a way the order of processing of data packets of different active connections is not strictly dependent on packet arrival order, but it is controlled by active program scheduler.

## 3 Implementation

We are working on implementation of the architecture described above using software routers based on the NetBSD (UNIX) operating system and off-the-shelf IBM PC-compatible hardware. Such routers are widely used in our metropolitan area network and they represent a general programmable platform suitable for implementation of novel network architectures and protocols. In addition to our previous experience, another reason to choose the NetBSD operating system was its well designed infrastructure for creating alternative OS personalities (emulations) which enables us to create (relatively easily) an execution environment where native binary (e.g. compiled C or C++) active programs can be securely executed.

While this operating system environment influences decisions about computer languages for implementation and the use of particular system functions, the active network model is not restricted in any way to this environment (i.e. the NetBSD based PC routers) and is implementable in any general enough computing environment.

Adapting a PC router to an active router means to introduce new features on the kernel and user levels. The most important components of our active router architecture are depicted in the Fig. 1. On the user level, the authentication and authorization module must be added, together with the full environment for the session and packet processing programs including parts of the resource management and statistics collection.

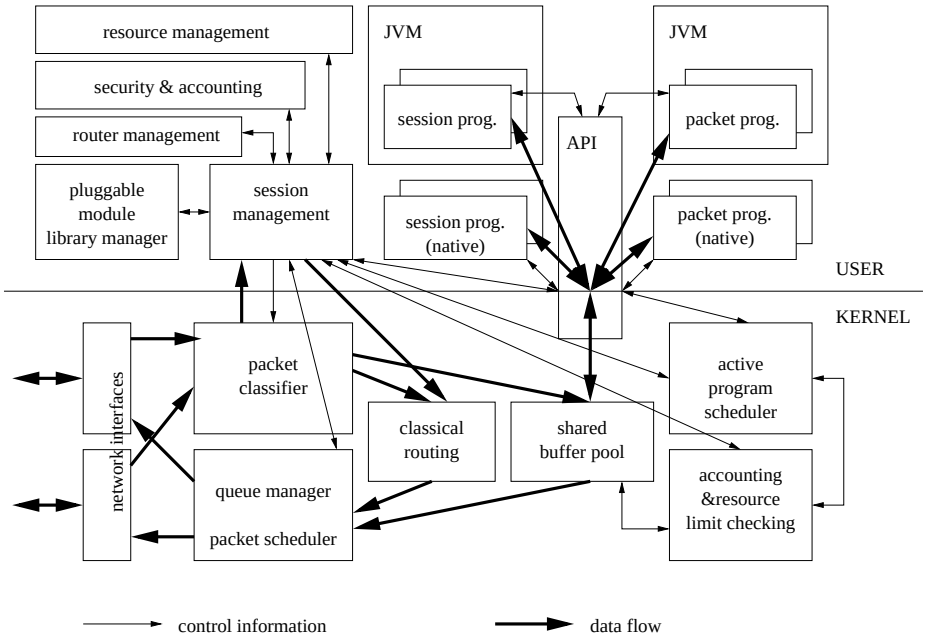
On the kernel level, it is necessary to create restricted execution interface for secure execution of native binary active programs, to add more general interface to packet classification code (standard part of PC router) which recognizes data packets of active connections, and also to add some changes to support more fine-grained scheduling for active programs.

Our implementation uses the C language to implement both the user and kernel level modifications and extensions of the PC router. While this means that the implementation will take more time then using high level approach with Java or similar programming environment (the implementation times compared in [4] are 3 years against 6 months), the result should be much more efficient. We believe that highly efficient implementation is necessary if we are to show the advantages of active networks.

## 4 Related work

During the relatively short history of active networks research, several models or network architectures have been developed. These models can be divided to two basic groups based on means by which user code is supplied to programmable elements of the active network. The first group places the code of active program inside the data packet that will be processed by the program ("active packet" or "capsule"). The other approach ("active nodes") uses other ways of program distribution, e.g. preloading of fixed function sets active nodes, on-demand program downloading, distributed caching of programs, etc. For example, typical

**Fig. 1.** Active Router Architecture




representatives of the first group are *Smart packets* project [7], *ANTS* [8], and *PAN* [9]; representatives of the second group are *SwitchWare* [5] or *Pronto* [6] systems. Both these two architectures have specific advantages and could be combined in one model.

Pronto is quite similar to our implementation of active node operating system support, in fact it could serve as alternative base for implementation of the architecture presented. The most important difference is lack of support for untrusted binary code execution in Pronto (user code interfaces with C++ class library).

The SwitchWare architecture also has some features in common with the model we proposed, e.g. support for multiple programming languages for user code. On the other hand, in SwitchWare, active packets always carry code in addition to data and this code cannot be completely preloaded on active node, unless it is defined as extension of active router.

## 5 Conclusion

The active network protocol design and its implementation within PC routers was motivated by the work in the area of quality of services for multicast, and voice and video applications in connectionless networks [3]. The active network technology looks promising for solving these and related problems and it also

provides convenient environment for research, where different QoS and/or multicast protocols can be implemented and tested.

## 6 Acknowledgements

The authors would like to thank Luděk Matyska for support of our work, proof reading this text and stimulating discussion about the quality of services in computer networks now and in the future.

The work was supported by the CESNET Research intent MSM000000001, and one of the authors (EH) also acknowledge support from Universities development fund grant No:0430/2001.

## References

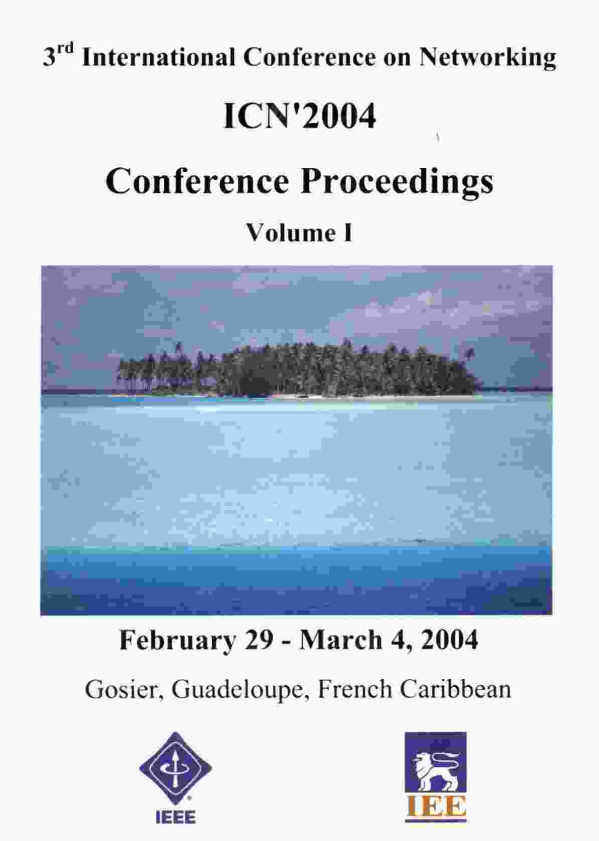
1. D.L. Tennehouse, J.M. Smith, G.M.W. Sincoskie, D. J. Wetherall and G. J. Minden. *A survey of active network research.* IEEE Communications Magazine 35, 1997.
2. K. Psounis. *Active networks: Applications, security, safety and architectures.* IEEE Communication Surveys, 1999.
3. D. J. Wetherall et.al. *Introducing New Internet Services: Why and How.* IEEE Network, May/June 1998.
4. J. P. Gelas, L. Lefevre. *TAMANOIR: A High Performance Active Network Framework.* Workshop on Active Middleware Services 2000 at the Ninth IEEE International Symposium on High Performance Distributed Computing, Pittsburgh, Pennsylvania, USA, August 2000.
5. D. Scott Alexander, William A. Arbaugh, Michael W. Hicks, Pankaj Kakkar, Angelos D. Keromytis, Jonathan T. Moore, Carl A. Gunter, Scott M. Nettles, and Jonathan M. Smith. *The SwitchWare Active Network Architecture.* IEEE Network Special Issue on Active and Controllable Networks, vol. 12 no. 3, pp. 29–36.
6. G. Hjálmtýsson. *The Pronto Platform: A Flexible Toolkit for Programming Networks using a Commodity Operating System.* In IEEE OPENARCH 2000, March 2000.
7. Schwartz, Beverly I., W. Zhou, A. W. Jackson, W. T. Strayer, D. Rockwell, and C. Partridge. *Smart Packets for Active Networks.* Proceedings of InfoComm, New York, 1999.
8. D. J. Wetherall, J. V. Guttag and D. L. Tennehouse. *ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols.* In IEEE OPENARCH'98, April 1998.
9. E.L. Nygren, S.J. Garland and M.F. Kaashoek. *PAN: A High–Performance Active Network Node Supporting Multiple Mobile Code Systems.* In IEEE OPENARCH'99, March 1999.

# Appendix B

# User Empowered Virtual Multicast for Multimedia Distribution

by Eva Hladká, Petr Holub and Jiří Denemark

In The Third International Conference on Networking, ICN 2004, Gosier, Guadeloupe, French Caribbean, March 2004. Proceedings. 2004. pp. 338–343. ISBN 0-86341-325-0.



# User Empowered Virtual Multicast for Multimedia Communication

Eva Hladká
Laboratory of Advanced Network Technologies
Faculty of Informatics,
Masaryk University Brno,
Botanická 68a, Brno 602 00, Czech Republic
Email: eva@fi.muni.cz Phone: +420 549 493 535

Petr Holub
Faculty of Informatics and
Institute of Computer Science
Masaryk University Brno,
Botanická 68a, Brno 602 00, Czech Republic
Email: hopet@ics.muni.cz Phone: +420 549 493 944

Jiří Denemark
Labor. of Advanced Network Technologies
Faculty of Informatics,
Masaryk University Brno,
Botanická 68a, Brno 602 00, Czech Republic
Email: jirka@ics.muni.cz

***Abstract*— We introduce concept of user empowered UDP packet reflectors to create virtual multicasting environment as an overlay on top of current unicast networks. The virtual multicast is used as a bottom layer for secure and efficient collaborative environments. The end-users' ability to fully control this environment—in a way similar to peer to peer networks—is the primary advantage of our approach. Other interesting features that are possible only in virtual multicast environment are also discussed in this paper.**

***Index Terms*— virtual multicast, UDP packet reflector, user empowered approach, modular architecture**

## I. Introduction

Multicast is the "natural" solution for a group communication [1]. Multicast communication can be characterized by the following statement: "The same data are transferred at most once on any particular link". This implies large ("infinity") scalability, but imposes non-trivial requirements on the network as all the network nodes must support it in a consistent way. Despite continuous effort only very small fraction of places on Internet have reliable native multicast connectivity. While the radio, television, and other mostly one-way broadcasting systems are practically impossible to be deployed on large scale without native multicast support, the collaborative environment usually connects bidirectionally few places only and "infinity" scalability is not such pressing issue. The communicating groups have usually at most 20 sites connected while larger groups need very precise orchestration and moderation. All the practically used multicast protocols have also other disadvantages: it is near to impossible to take care of quality of service requirements for the whole multicast group, it is very difficult to provide secured environment without a shared key, and there is no easy support for accounting.

This research is supported by a research intent "High Speed Research Network and its New Applications" (MSM000000001).

These problems may be overcome through multicast connectivity simulation, where active nodes have a role of *reflectors* ("mirrors"), that replicate all traffic passing through them in a controlled way. In such environment multicast videoconferencing clients can be used with ease while keeping the advantages of unicast point-to-point communication lines—thus creating *virtual multicast environment* (this technology is used e. g. in VRVS [2] or AccessGrid [3]). The reflectors can even transform the incoming traffic and can be directly controlled by the end users. These mirrors play a role of multicast join-points, allowing clients to connect (and drop out) without any undesired influence on the rest of the group.

We propose reflector architecture based on active router architecture [4]. This architecture can be used for creation of *ad hoc* overlay networks, where both mirrors and the overlay network creation is administered directly by end users. The behavior of each individual mirror can be independently controlled, including the security environment. The security context may be individualized for each client, using any authentication scheme, including PKI, Kerberos or shared keys. While reflector technology is only partially scalable it is the most efficient infrastructure for groups with no more than tens of clients. The main advantage is flexibility, user empowered-ness, and independence on any specific network features except for simple unicast routing. All the "advanced features" are provided by higher, user controlled layer. Any user group can start its own mirror and only unicast connectivity is required from any client to the mirror. Two or more mirrors may be combined to provide a true overlay network. While data routing and replicating are the basic functions, many more services can be provided within this

framework. Varying demands of different users' groups and even specific demands of individual users within a group can be handled by specific extensions (modules) to the basic mirror program in the active network framework. Few examples of already implemented features are: full logging and data recording, data encryption and decryption, synchronization of streams, authentication, authorization and accounting, and stream traffic shaping.

## II. Reflector architecture

The design of a reflector must be flexible enough to allow implementation of required features and leaving space for easy extensions for new features. This leads to a design that is very similar to our active router architecture [4] modified to work entirely within the user space. Users without administrator privileges are thus able to run reflector on any machine they have access to. The reflector architecture is shown in Fig. 1.

### A. Data routing and processing

Data routing and processing part of the reflector comprises *network listeners*, *shared memory*, *packet classifier*, *processor scheduler*, number of *processors*, and *packet scheduler/sender*.

Network listeners are bound to one UDP port each. When packet arrives to the listener it places the packet into shared memory and adds reference to a *to-be-processed queue*. The packet classifier then reads packets from that queue and determines a path of the data through the processor modules. It also checks with routing AAA module whether packet is allowed or not (in the later case it simply drops that packet and creates event that can be possibly logged). Zero-copy processing is used in all simple processors (packet filters), minimizing processing overhead (and thus packet delay). Only the most complex modules may require processing that is impossible without use of packet copies.

The *session management* module follows the processors and fills the distribution list of the target addresses. The filling step can be omitted if data passed through a special processor that filled the distribution list structure and marked data attribute appropriately (this allows client-specific processing). Processor can also poll session management module to obtain up to date list of clients for specified session. Session management module also takes care of adding new clients to the session as well as removing inactive (stale) clients. When new client sends packets for the first time, session management module adds client to the distribution list (data from forbidden client has already been dropped by packet classifier). Information about the last activity of a client is also maintained by the session module and is used for pruning stale clients periodically. Even when distribution list is not filled by the session management module, packets must pass through it to allow addition of new clients and removal of stale ones.

When the packet targets are determined by the router processor a reference to the packet is put into the *to-be-sent queue*. Then the packet scheduler/sender picks up packets from that queue, schedules them for transmission, and finally sends them to the network. Per client packet scheduling can also be used for e. g. client specific traffic shaping.

The *processor scheduler* is not only responsible for the processors scheduling but it also takes care of start-up and (possibly forced) shutdown of processors which can be controlled via administrative interface of the reflector. It checks resource limits with routing AAA module while scheduling and provides back some statistics for accounting purposes.

### B. Administrative part of the reflector

Communication with the reflector from the administrative point of view is provided using *messaging interfaces*, *management module*, and *administrative AAA module* of the reflector. Commands for the management module are written in a specific *message language*.

Messaging interface is generic entity, which can be instantiated as e. g. RPC, SOAP over HTTP, plain HTTP interface with SSL/TLS or GSI support, or simple TCP connection bound to loop-back interface of the machine running the reflector. Each of these interfaces unwraps the message if necessary and passes it to the management module.

Management module checks validity of the message and its authenticity and authorization status, querying the administrative AAA module, which is also responsible for the processing of accounting information of the accepted messages. Availability of sufficient resources to process accepted message is checked with resource management module and the message is passed to the appropriate module(s) afterwards. Completion status is returned back to the management module, which notifies the administrative AAA module to close the accounting. If a failure occurs, its description is stored via the administrative AAA module, an error that can be logged is generated, and an error message is simultaneously sent back the client via messaging interface the client is connected to.

The same mechanism can be used for *logging* purposes. One or more messaging interfaces may be opened with the `LOGGING` flag set and such messaging interfaces receive all events created within the reflector via the management module. The way how to receive logging information through some messaging interface, which doesn't have the `LOGGING` flag set, is to send a request asking for logging information via this interface.

A message language for communication with the management module is called Reflector Administration Protocol (RAP) [5]. It is a text-based request/response protocol that uses US-ASCII character set. Lines are delimited by character pair CR and LF (0x13 and 0x10). Protocol message can be either user's request to the reflector or response of the reflector to the user. One request can be followed by multiple responses. Protocol is designed as soft-state, i. e. connection is closed after certain period of client inactivity. Keep-alive messages have to be sent if the client wants to maintain connection and has no requests to send.

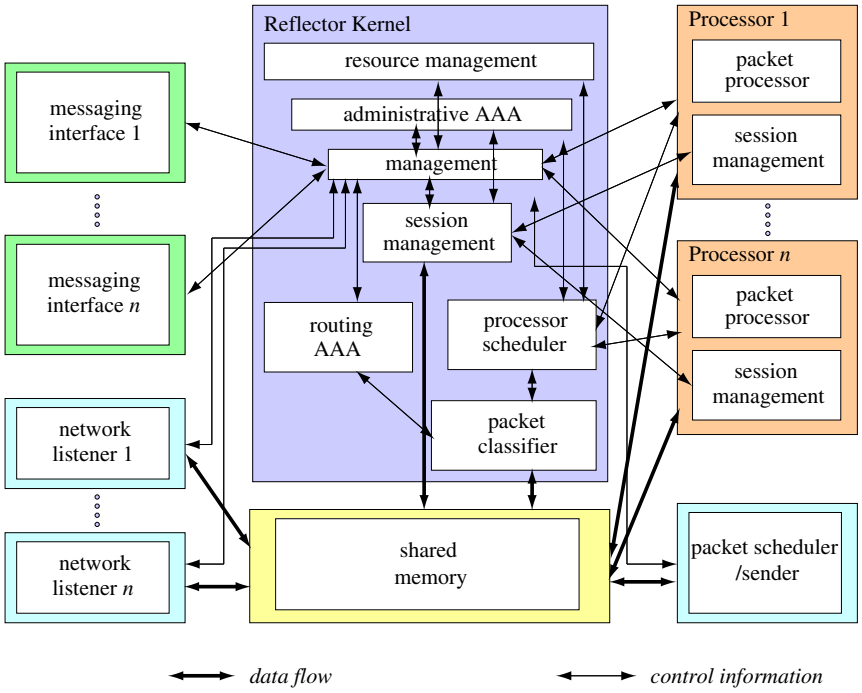



Fig. 1. Reflector Architecture

Each message comprises message headers and message body. In case of request, the headers contain

- specification of method (that is an actual command for the reflector) and method specific headers
- target module reference (RAP contains a module addressing schema enabling to pass requests to specific modules),
- protocol version identification, length of the message body, possible request to process the request in blocking manner, and other auxiliary information

Message body contains method specific information if needed. Methods (or commands) can be divided into two groups: general reflector methods and extending module specific methods. Group of general methods covers methods for requesting various information on status of the reflector and its modules (either in interactive way or in form of subscription for logging information), commands for manipulating both routing and administrative access control lists as well as administrative user database, and commands for controlling modules (starting, stopping, and restarting modules).

There are several response classes used by reflector to reply to client's request: 100 (for informational and logging messages), 200 (for successful completion of user request with message body containing actual response data), 400 (for client request errors), and 500 (for server-side error).

Complete definition of RAP version 1.0 including formal definition using ABNF [6], detailed protocol description, and example communication using RAP can be found in [5].

## III. Advanced Reflector features

The basic function of the reflector is retransmission of received data to one or more listeners. This can be easily extended to support other useful functions. The reflector replicates all the traffic coming through specified port to all the clients connected to that port. Clients do not interact in advance—they just connect to the reflector to automatically receive all the traffic sent to the reflector and also all the client traffic is automatically distributed by the reflector. The reflector security (per port or per client) policy may change this behavior and forbid some clients from listening or sending data.

*a) Tunneling and scalability:* The scalability of the reflector based communication can be increased using tunneling between the reflectors. Possible tunnel configurations are:

- *full-mesh tunneling* – Each reflector has addresses of its peers and sends data received from directly connected clients to all the peer reflectors and accepts data from all the peer reflectors and distributes it to all directly connected clients. This is the least efficient way with respect to scalability but has the simplest setup.
- *static tunneling* – Routing among reflectors is done by manually pre-configured static way. Such tunneling is

used for MultiSession Bridge in AccessGrid[7]. This can be efficient for long-lived network of reflectors with infrequent changes. It is not suitable for networks created in *ad hoc* mode. When properly managed, this is the most efficient configuration.

- *dynamic tunneling and routing* – Mimics the behavior of routers and may use very specific routing algorithms (even multi-criteria). The simpler configurations may use distance vector routing algorithms used in `mrouted` (on the MBone) or even algorithms from peer-to-peer networks might be used (e. g. an algorithm developed in Pastry project [8]). This option is the most suitable for *ad hoc* networks of reflectors and may be the most efficient for dynamic environments.

Both static and dynamic configurations are suitable for networks with some low bandwidth or high latency lines since they can be configured to bypass such links.

*b) Logging:* Following events are examples of events generated within the reflector either as a result of data transmission or reacting to managerial decisions: start-up and shutdown of the reflector, beginning and end of data transmission, start and stop of data recording, users' login and log-off times and login failures, non-authenticated requests to join a group, program errors. These events are sent to all the messaging interfaces subscribed for logging information, which store them into disk files or pass them on e. g. the system `syslog` interface or to some external monitoring tools.

*c) Stream transcoding:* A specific module can act as a multimedia stream transcoder. This is usually used when some client is connected via the low bandwidth link. As the transcoding consumes rather lot of processing power, a specific reflector—the *gateway*—may be set up at the beginning of the low bandwidth link and connected to the nearest reflector using the tunneling capability and thus enables clients connected via low bandwidth links to participate without influencing clients connected using fast links.

Another situation in which the transcoding feature is useful is when media stream is produced and transferred in format that is not acceptable for the some client(s).

*d) Video stream composition:* In some circumstances, like different quality of links used by participants, large groups resulting in too many windows at client sites, insufficient processing power at client sites to decode large number of simultaneous streams etc., it may be advantageous to down-sample video streams and compose several of them into one stream directly on the reflector (provided it has sufficient memory and processing power). The first version of such a processor is described in [9] and it has already been adapted to fit into the new architecture. This processor is based on the `vic` tool and support exactly the same set of video formats. Up to four video streams can be composed into one output stream. Input video formats are auto-detected, the processor is able to work with different formats simultaneously. The output video format is configurable by the end user.

*e) Recording:* The centralized data recording facility has many advantages over much simpler features of most videoconferencing clients. It may be a trusted neutral agent (no local editing of content), it can guarantee to store all data actually transmitted (in contrast to local client which may experience some local data loss), it may be more efficient (just one copy of the data is created), and it provides recording capabilities independent of whether the client software has recording capabilities or not. The recording processor within the reflector is controlled by end-users, stored data can be easily used to re-play the communication. Storage of all the data transmitted allows a synchronized re-play of not only audio or video but also of shared workspace modifications, too.

*f) Synchronization:* Several independent UDP streams may be synchronized using the appropriate processor. It inspects timestamps and delays or reorders packets within a specified time windows so the resulting outbound streams are fully synchronized. It uses timestamps in RTP packets and is able to work with independent time references ("zero time") and increments for each stream (even different streams from the same source computer can use different time references [10], [11]). The information needed for converting between real (absolute) time and relative RTP time is taken from RTCP packets. This requires sending computers to have their real time clocks synchronized e. g. using NTP protocol.

This synchronization feature can be used e. g. to send 3D video in two streams for each eye separately synchronously over best effort network. Such transmission allows to use more demanding video processing compared to sending it in one stream since the processing can be distributed among two or even more machines provided they have their real time synchronized properly.

A preliminary implementation, which uses dedicated reflector lacking our new modular architecture, has been described in [12].

*g) Traffic Shaping:* The traffic shaping processor supports among other: bandwidth limiting, delaying, deliberate packet loss, and packet duplication (on the same stream). The last two features are used usually for debugging purposes or for simulation non-ideal network conditions. Delaying is used to increase fairness among videoconferencing partners in unfair conditions (where one or more partners have substantially larger delays).

*h) Raw Data Dumping:* While events from management module are stored via the logging interfaces, the reflector supports also raw data dumps of all incoming packets (including those rejected for authentication or authorization reasons). The reflector, running in the user space, does not rely on the `tcpdump` (requiring root privileges) and in the dumping mode it simply copies all data on inbound interfaces into a file for later analysis.

## IV. Security in context of the reflector

The reflector is a program started by an ordinary user—who thus becomes primary reflector administrator—and it

runs under his or her identity. As this user grants some privileges to partners within the group, the reflector must protect user from malicious behavior of third parties. This is done via authentication and authorization mechanisms that are part of the administrative AAA module. In various scenarios (e. g. military or bioinformatics) the actual data communicated among partners must be protected as well.

All the administration is done via secure messaging channels (e. g. SSL/TLS secured HTTP). User can authenticate using login and password or via some authentication credential (e. g. Kerberos ticket or X.509 certificate). The authorization is done using ACL (access control list) and is performed per command (similar to the TACACS authorization mechanism). The reflector administrator creates the first ACLs and also specifies (during compile-time and reflector startup) which authentication mechanisms will be supported.

*i) Basic end-user security:* Simple client authorization is based on IP address restrictions. Appropriate "accept" and "deny" ACL records contain IP addresses or subnets (defined as an IP addresses with associated netmasks). The decision is taken by the routing AAA module, rejected packets are dropped and appropriate event is generated. This decision precedes the session management phase to eliminate work that would be otherwise discarded. However, session management must be informed about changes in ACLs to be able to discard forbidden clients immediately.

Restrictions based on user names are done indirectly—a user connects via secure messaging channel and adds his/her IP address into the list of accepted IP addresses. In such scenario it is also possible to employ soft-state mechanism when certain IP address or address range is accepted over limited period of time only. The user is responsible to renew the authorization in regular intervals.

*j) Strong end-user security:* Areas like military and medicine require strong security support. This issue was studied in [13] and there is a solution currently available for the reflector.

The secure reflector consists of four parts: initialization, authentication, communication, client disconnection. The *initialization phase* processes the input parameters, initializes cryptography subsystem and waits for client to connect.

In *authentication phase* client and server set up secure channel using RSA secured TCP connection. The client authentication is then based on user login and password. When accepted, the secure connection is maintained during the whole conference as it is used for session tear-down and optionally for redistribution of keys if temporal re-keying is enabled.

In *communication phase* reflector forwards data encrypted using symmetric AES cipher using key exchanged during authentication phase with clients.

In the final *client disconnection phase* client asks for end of session and is removed from the list of allowed clients by the server. Disconnection phase can also be initiated by the server when it shuts down.

Client side is realized by specialized local reflector which client tools (e. g. `vic` or `rat`) connect to[1]. This local reflector processes the authentication, exchanges session keys with the reflector (each client/server pair has its own session key) and is responsible for data encryption and decryption using these session keys.

*k) Use of reflectors in adverse networking environments:* *Firewalls* are spread in many places and are the administrative solution to protect LANs and their resources from malicious users and accidents. This protection has a negative side effect since it means barrier to free network communication and makes difficulties when deploying applications relying on user empowered paradigm in case when the firewall is not controlled by the end user. It is usually difficult to achieve reconfiguration of firewall for communication on unusual ports. Another problem that often goes hand in hand with firewalls is *network address translation (NAT)*. It usually doesn't work for TCP connection initiated from outside of inner network and for UDP traffic directed inside, the latter of which is a serious problem for virtual collaborative environments that rely on bidirectional UDP traffic for multimedia content.

A possible solution for reflector based communication may work without touching firewall configuration at all and it may also solve problems with NAT. First, we need to use two reflectors—one inside the zone protected by firewall and/or NAT and the second one outside. Reflectors need to have special processor that performs encapsulation of packets to pretend that they belong to some well known protocol that is passed through by most of the firewalls (HTTP being a good example). Our experiences show that this scenario can be deployed quite successfully[2] [14]. Such scenario works fine unless both clients are hidden in different networks performing NAT. In such case they need some reflector on outer public network that forms a kind of rendezvous point toward which both communicating clients point the tunnels of their local reflectors. Otherwise it may be impossible to address each other directly.

The client reflector is the one, which initiates the TCP connection with the desired address and port number of main reflector because of possible NAT, while the main reflector outside of the firewall waits and accepts the incoming TCP connections. RTP packets are sent through the TCP connection with minimal additional header prepended. The header consists of two 4 byte numbers, one carries the RTP packet length and the other is a flag distinguishing RTP data packets from RTCP packets.

## V. Future work

The work we are targeting now is to implement all described features to the reflector. For the future we plan to

[1] Some of client MBone tools require a small modification to be able to connect to the reflector running locally as they bind on the same port they are connecting to obviously resulting in conflict in such setup.

[2] The packet reflector modified for communication through firewall was tested successfully with voice communication in challenging environment (two wireless LAN hops and application layer firewall) between ICN'01 conference in Colmar, France, and Masaryk University in Brno, Czech Republic.

continue to support pilot user groups to get stimulating reactions and new ideas as this reflector has always been developed for group communication of real groups of users and lots of features are based on their demands. The most active are Czech group participating in EU DataGrid (EDG) project and IPv6 working group of the Czech research network.

For the more remote future we think about integrating our reflector into Open Grid Services Architecture framework [15] to enable integration with new generation of collaborative Grid environments (e. g. AccessGrid version 2.x [3]). Such integration requires incorporation of Grid service interfaces and Grid security mechanisms into our reflector.

As already mentioned in the section on tunneling between reflectors, another interesting direction of reflector development is implementation of self-organizing and automatic discovery capabilities stemming from ideas of peer-to-peer networking either in pure or hybrid or super-peer mode [8], [16]. This will enable reflectors to create overlay networks that can sustain even partial network disintegration without completely breaking overlay network as it allows the networks of reflectors in the remaining network clouds to work autonomously.

Furthermore we are considering extension of traffic shaping features and congestion control and development of more advanced data transformation processors.

## VI. Acknowledgment

The authors would like to give credits to Assoc. Prof. Luděk Matyska for supporting our work, proof reading this paper, and for stimulating discussions.

## References

[1] R. Wittmann and M. Zitterbart, *Multicast communication: protocols, programming, and applications*, Morgan Kaufmann Publishers, 1999.

[2] P. Galvez, G. Denis, and H. Newman, "Networking, Videoconferencing and Collaborative Environments," in *Proceedings of CHEP'98*, Chicago, September 1998.

[3] L. Childers, T. Disz, R. Olson, M. E. Papka, R. Stevens, and T. Udeshi, "Access Grid: Immersive Group-to-Group Collaborative Visualization," in *Proceedings of Immersive Projection Technology*, Ames, Iowa, 2000.

[4] E. Hladk´a and Z. Salvet, "An Active Network Architecture: Distributed Computer or Transport Medium," in *Proceedings of ICN 2001*, Berlin, 2001, vol. LNCS 2094, Springer-Verlag.

[5] J. Denemark, P. Holub, and E. Hladk´a, "RAP - Reflector Administration Protocol," Tech. Rep. 9/2003, CESNET, 2003.

[6] D. Ed. Crocker and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," RFC 2234, November 1997.

[7] G. A. Roediger, "The Multi-Session Bridge," `http://www.hep.net/chep98/PDF/230.pdf`.

[8] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, 2001, pp. 329–350.

[9] V. Holer, "Videostram Merging," Bc. Thesis FI MU, 2003.

[10] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 1889, January 1996, Obsoleted by RFC 3550.

[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003.

[12] T. Rebok and P. Holub, "Synchronizing RTP Packet Reflector," Tech. Rep. 7/2001, CESNET, 2003.

[13] Tom´aš Bouček, "Kryptografi ck´e zabezpečen´ı videokonferenc´ı," M.S. thesis, Military academy Brno, 2002.

[14] Z. Salvet, "Enhanced UDP packet reflector for unfriendly environments," Tech. Rep. 16/2001, CESNET, 2001.

[15] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.

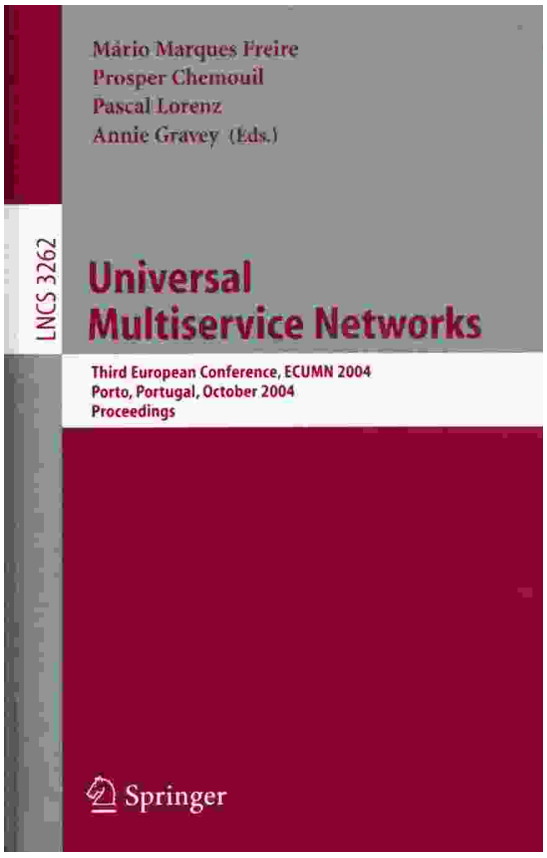
[16] B. Yang and H. Garcia-Molina, "Designing a super-peer network," IEEE International Conference on Data Engineering, 2003.

# Appendix C

# User Empowered Programmable Network Support for collaborative Environment

by Eva Hladká, Petr Holub and Jiří Denemark

In Universal Multiservice Networks: Third European Conference, ECUMN 2004, Porto, Portugal, October 25-27, 2004. Proceedings. Lecture Notes in Computer Science 3262, Heidelberg: Springer-Verlag Berlin, 2004. 10 p. ISSN 0302-9743.



# User-Empowered Programmable Network Support for Collaborative Environment

Eva Hladká[1], Petr Holub[1,2], and Jiří Denemark[1]

[1] Faculty of Informatics
[2] Institute of Computer Science,
Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
eva@fi.muni.cz, hopet@ics.muni.cz, xdenemar@fi.muni.cz

**Abstract.** We introduce a user-empowered UDP packet reflector to create virtual multicasting environments as an overlay on top of current unicast networks. The end-users' ability to fully control this environment by a specific communication protocol is the main advantage of our approach. Serializing the parallel communication schema for group communication allows us to introduce special features that are possible in unicast communication only. Similar to working with programmable routers, users can submit their own modules, which can be linked into the reflector and perform user-specific operations (filtering, transcoding etc.). The reflector is the basic element of the overlay network support for the user-empowered group communication in collaborative environments.

## 1 Introduction and Theoretical Background

In the current world, people are looking for systems supporting easy-to-use and inexpensive activities like video-seminars, tele- and video-consulting, and virtual meetings, which are specific forms of a virtual collaborative environment [1]. This paper focuses on both building a theoretical framework and creating a practical implementation of a network support system for communication among smaller groups of participants (up to 20 sites, usually bellow 10) that can be fully controlled by the participants themselves. The system is intended to be simple to use and yet flexible, capable of reacting to pre-defined as well as dynamic events such as changes in number and location of participants, network conditions (bandwidth, delay, security), etc.

Two basic principles are being adopted in a complementary way when transferring data over the networks: *connection oriented* and *packet oriented* approaches. Both approaches reached widespread use in different environments and nowadays we see a lot of effort dedicated to their convergence. This is also dictated by new applications and their requirements of scalability on one hand and transport quality control on the other hand. The packet based networks with rather "dumb" active elements targeted for only one function—data routing—won the field of high-speed networks, while sacrificing most of the control features needed for advanced applications. A quality of service is offered on a statistical

basis only (e. g. using DiffServ approach) and the users usually have no way of influencing or at least monitoring the transport of "their" data over the network. As reaction to these problems, we are developing a novel approach based on following cornerstones:



- *Active elements* within the network, programmable directly or at least indirectly by the users and their applications. These serve as the underlying technology for implementing the higher layers [2].
- *Overlay networks* as a framework for introducing specific services within the packet oriented network. The overlay networks allow minimizing the necessary overhead for advanced services without limiting their complexity [3].
- *User empowered approach* as a way to put the control plane into the hands of end users. The users can set up and tear down services, control and monitor their behavior while the services are well isolated so as to avoid any unwanted influence on other users.

The reflector is built as a special active node within a network, with full control by the user who uses it for group communication. The active router was modified to serve as the user controllable (user empowered) multimedia data reflector. The active node is implemented as a specific service within an ordinary computer. Fulfilling the requirement of full user control means overworking the active router and moving its functionality into the user space without any changes on the kernel level. This special implementation of an active router in user space was created and became the basic element for the overlay network for group communication.

## 2 Reflector

The *reflector* is a network element that replicates and optionally processes incoming data usually in the form of UDP datagrams and distributes this data to its clients in sequential manner using unicast communication only. If the data is sent to all listening clients, the number of data copies is equal to the number of clients. Our reflector is designed as a user-controlled modular programmable router, which can optionally be linked to special processing modules in run-time. The reflector runs entirely in user-space of the underlying operating system and thus it works without the need for administrative privileges on the host computer. The reflector architecture comprising the administrative part and data routing and the processing part is shown in Fig. 1.

The data processing and replication works as follows: the entry points of the reflector are network listener modules which are bound to one UDP port each. The received packet is placed into the shared memory and the listener adds a reference to a "to-be-processed" queue. A packet classifier reads the packets from this queue, checks with a routing AAA module whether the packet is allowed or not and determines a path through processor modules for each packet. After the processing, the data is distributed to clients by a packet scheduler/sender module according to a distribution list obtained from a session management module. The

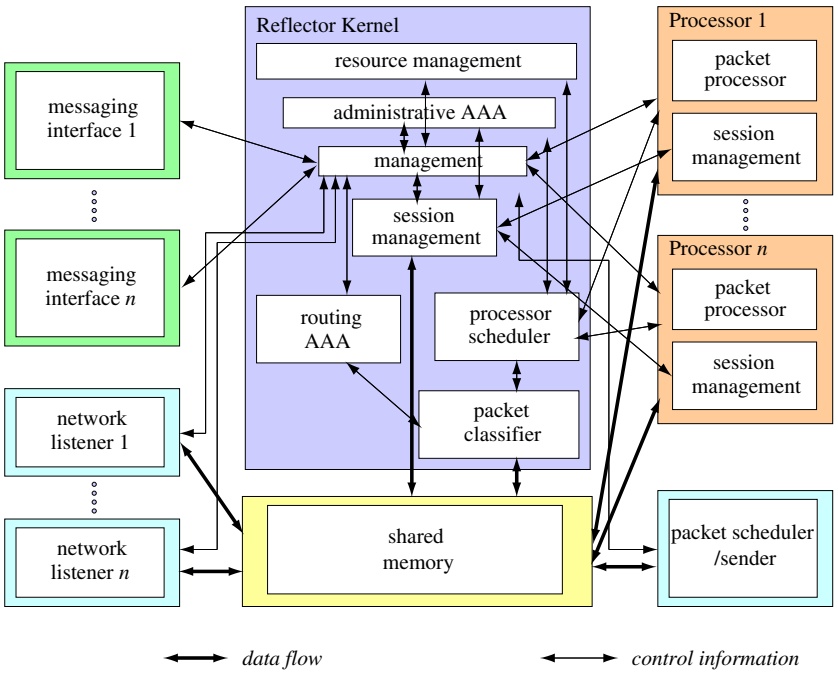



**Fig. 1.** Architecture of the reflector.

number of copies of the data inside the reflector is minimized in order to boost performance; for simple scenarios the reflector works in the zero-copy mode.

The session management module is responsible for maintaining the distribution lists for each group, for adding new clients (usually after the client starts to send data), and removing inactive (dormant) clients. Simple client authorization is based on IP address restrictions. The access control list contains an "accept" or "deny" rule for each IP address or subnet record. The decision is made by the routing AAA module, rejected packets are dropped and an appropriate event is generated and can be logged if requested.

The administrative part of the reflector can be accessed via secure messaging channels such as HTTP with SSL/TLS encrypted transport or SOAP with GSI support[3]. The user can authenticate using various authentication procedures, e. g. combination of login and password, Kerberos ticket, or X.509 certificate. Authorization uses access control lists (ACLs) and is performed on per-command basis. Authentication, authorization, and accounting for the administrative part of the reflector is provided by an administrative AAA module. The actual reflector control is provided by a management module, which accepts commands in a specific messaging language, the Reflector Administration Protocol (RAP) [4].

[3] Basic transport used for secure web services in Grid environments. `http://www.doesciencegrid.org/Grid/projects/soap/`

All the events that occur in the reflector (users joining or leaving the reflector, exceptions etc.) can be logged for further inspection.

The data received by the reflector are replicated and sent back to all the connected clients and thus the limiting outbound traffic on the reflector grows with $n(n-1)$ where $n$ is the number of active (sending) clients. The scalability issue arises obviously which can be mitigated by creating networks of reflectors with tunnels connecting them (see Sec. 3.1). The network can be built in either a static way (pre-configured) or dynamic way (e. g. using distance-vector routing algorithms or some more efficient routing algorithms from peer-to-peer networks like Pastry [5]). Reflector networks can also be used for building overlay networks that are more resilient to network outages than the underlying network [3].



### 2.1 Advanced Reflector Scenarios

Because of the data replication for each individual client, it is possible to implement per-user processing which is impossible to do with multicast. The modularity of the reflector allows users to add and configure specific functionality in run-time. Examples of per-user processing are shown below:

- *Multimedia data transcoding.* Data processing modules can convert data between different formats (e. g. re-compress data from the DV format to the H.261 format). The reflector can be thus used as a gateway allowing clients with limited support of compression formats or insufficient network or processing capacity to join videoconference without forcing the rest of the communicating group to use low-quality or low-bandwidth multimedia formats.
- *Video image composition.* Composing several video images into a single image can be useful for a collaborative environment with a large number of participants in which there is not sufficient processing or display capacity to provide full video windows from all the clients simultaneously.
- *Synchronization.* When using parallel media streams encapsulated in RTP protocol, it is possible to synchronize such streams [6]. RTP packets contain relative time-stamps that can be converted to absolute local time on the sending machine by utilizing both relative and absolute time-stamps sent in complementary RTCP packets. When the synchronized streams originate on different computers, it is necessary to synchronize time on these computers, e. g. using NTP protocol.

By connecting reflectors with different functionality, it is possible to create an overlay network allowing users to connect to reflectors according to their needs.

Reflectors can be used for building strongly secured communication and collaboration environments. In the secured scenario each client must maintain a secured reliable connection to the reflector (usually an SSL encrypted TCP connection) that is used to exchange encryption keys between the client and the reflector. UDP datagrams are then sent encrypted from the client to the reflector, processed, and distributed to other clients encrypted again. Such reflectors however, requires modified MBone Tools to work with [7]. The reflector can also

be used in an adverse networking environment restricted by firewalls and NAT deployment since it is possible to tunnel UDP data between reflectors through a TCP connection using some well-known ports that are enabled on the firewall [8].

C

## 3 Performance Evaluation of a Prototype Implementation

The reflector described above has been implemented and its performance has been evaluated in order to verify its usability. The testbed environment comprised three powerful machines used as a traffic generator (`gerard`), a reflector (`test4`), and a receiver (`brand`). The machines were connected via the HP ProCurve 6108 gigabit Ethernet switch. More detailed information on configuration of these machines is shown in Tab. 1.

**Table 1.** Overview of configurations of the testbed machines.

| | `test4` | `brand` | `gerard` |
|---|---|---|---|
| *brand* | – | DELL PowerEdge | DELL PowerEdge |
| *model* | – | 1600 SC | 1600 SC |
| *processor* | 2× Intel Xeon 2.80 GHz | 2× Intel Xeon 2.80 GHz | 2× Intel Xeon 2.80 GHz |
| *memory* | 1024 MB | 1024 MB | 1024 MB |
| *NIC* | Intel 82545EM 64 bit/66 MHz | Broadcom BCM5701 64 bit/100 MHz | Intel PRO/1000 32 bit/66 MHz |
| *operating system* | Linux 2.4.23 | FreeBSD 5.2-RELEASE | FreeBSD 5.2-RELEASE |

To evaluate the performance, clients sending 30 Mbps stream each were used thus emulating multimedia clients utilizing DV [9] video format sent in RTP packets over the IP network [10]. During the experiment the number of active (both sending and receiving) clients was increased and there was a single passive (listening only) client used as a measuring probe. The results summarized in Fig. 2 show that the system is usable for communication of up to five active clients working with very high quality video. For clients with lower bandwidth utilization, the number of clients that can get connected grows $n \propto 1/b$ where $n$ is the maximum number of connected clients and $b$ is the bandwidth used. It is also obvious from the results that the reflector is capable of fully saturating a gigabit Ethernet network link with limits imposed by the hardware and operating system used. The problem of scalability can be further tackled by building networks of reflectors (Sec. 3.1).

We have also evaluated the maximum forwarding throughput of the reflector which proves to be more demanding compared to common replication. This corresponds to the fact that more data is transmitted over the PCI bus compared to the replication mode. The results summarized in Fig. 3 show that it is possible to forward approximately 450 Mbps without significant packet loss.

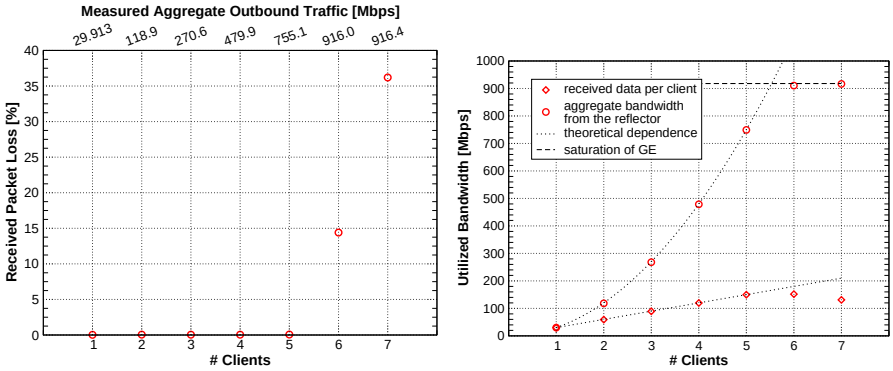



**Fig. 2.** Reflector prototype performance evaluation for 30 Mbps clients.

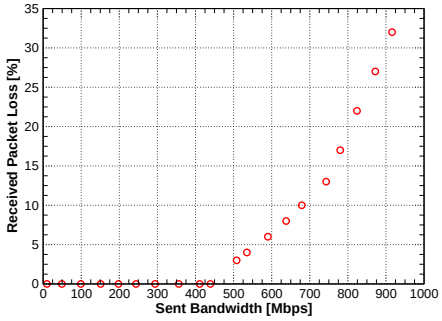



**Fig. 3.** Raw forwarding performance of the reflector.

### 3.1 Scalability Implications

As already mentioned in the Sec. 2 and 3, the scalability of the reflector-based communication environment can be further improved by creating networks of the reflectors connected via tunnels. The simplest model, which can also be used as the worst case estimate, is a complete graph in which each reflector communicates directly with all the remaining reflectors as shown in Fig. 4a. We call such model *full mesh tunneling*.

Let's assume a mesh of the reflectors in which each reflector has either $n_r$ or $n_r - 1$ clients resulting in the most balanced population of reflectors with clients. It is possible to show that the number of inbound streams on each reflector is

$$in = n, \tag{1}$$

where $n$ is the total number of clients. The number of outbound streams for reflector with $n_r$ clients is

$$out_{n_r} = n_r(m + n - 2), \tag{2}$$

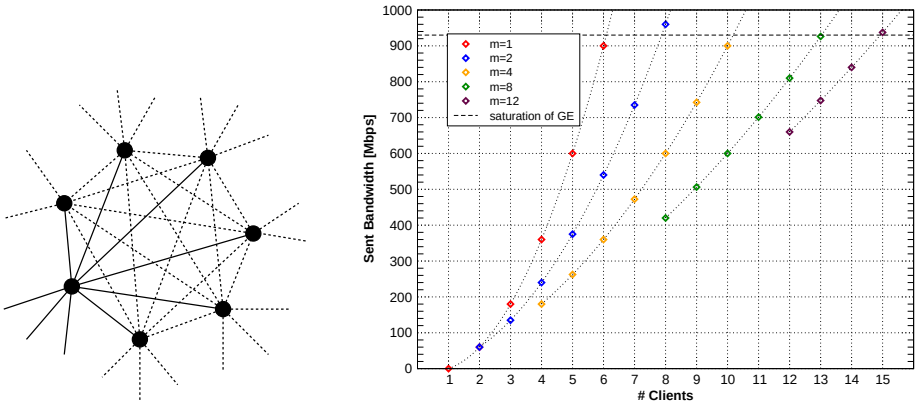
where $m$ is the total number of reflectors in the mesh. The ratio of outbound traffic for the reflectors with $n_r$ and $n_r - 1$ clients is

$$\frac{out_{n_r-1}}{out_{n_r}} = \frac{n_r - 1}{n_r}. \tag{3}$$

Taking into account that $n_r = \lceil \frac{n}{m} \rceil$ and the number of streams on a single stand-alone reflector is $out_s = n(n-1)$, it is possible to show that the ratio between the limiting outbound traffic for the reflector participating in the mesh of reflectors $out_{n_r}$ and a stand-alone reflector $out_s$ is

$$\frac{out_{n_r}}{out_s} \approx \frac{m+n-2}{m(n-1)}. \tag{4}$$

As illustrated in Fig. 4b for meshes with varying numbers of the reflectors, it is possible to increase the number of clients sending 30 Mbps to 15 when a mesh of 12 reflectors with gigabit network link is used.




(a) (b)

**Fig. 4.** (a) Full mesh reflector tunneling model. (b) Dependence of the number of 30 Mbps clients on the number of reflectors in the mesh.

## 4 End-User Applications

The reflector, capable of processing and distribution general purpose UDP packets, is able to support variety of end-user collaborative applications. Specific end-user requirements can be served by different versions, distinguished by specific processing capabilities (modules). Routinely, we use reflectors in connection with the `vic`, `rat`, and `wbd` tools from the MBone Tools package [11].

Different user groups used this environment during the last 3 years mostly for videoconferencing purposes, in very heterogeneous network conditions. While some groups enjoy rather homogeneous network environment, where all clients are connected through 100 Mbps network and the backbone runs on 1 Gbps and above speed, the reflector-based videoconferencing system is also regularly used in an environment where some clients are connected via cable TV. Even the network latency can be accommodated—we support a user group where most members are located in the Czech Republic (with clients both on high speed academic network and on a cable TV one) and one client connects via cable TV from Seattle (Washington, USA). Such an environment is rather hostile to the native MBone while reflector based data distribution works without problems. Another experimentally confirmed advantage is simultaneous support of different versions of MBone tools—again very problematic with native multicast.

As current high-quality videoconferencing tools tend to utilize high quality and high-bandwidth multimedia streams, we have also successfully tested the reflector with the DV over IP transmission tools from the DVTS project. We re-implemented the `xdwshow` tool (to overcome some problems encountered in its official implementation) and we have versions for PAL and NTSC formats under the Linux and FreeBSD operating systems. Our implementation uses robust thread architecture, where individual threads are used to input data from network, render them and display the resulting picture. This new implementation also support communication with reflector [12]. We plan to use this high quality video environment for teaching purposes, e.g. real time transmission from a neurological operation. Also, such high quality video streams can be used for the 3D video, using also the synchronization feature of the reflector.

## 5 Conclusion

The reflector is an active programmable network node providing all the necessary support for group communication in unicast networks. Our solution can simulate the multicast connectivity transparently, so the multicast clients can be used with ease, while keeping the advantages of unicast point-to-point communication lines. The reflectors function as multicast rendezvous points, allowing clients to connect and leave without any undesired influence on the rest of the group. The startup and shutdown of reflectors is a part of active network programming and as such it is fully user controlled. Users are also free to connect reflectors together in an *ad hoc* way and to specify behavior of each individual reflector, including possible security requirements and QoS parameters. A more general contribution is the method introducing new services into a network. The user empowered overlay network can be built a local scope (where needed), using only the features actually needed and within a limited time frame only. The network in this case is not overloaded by new protocols etc. and remains simple, robust, and fast.

While individual reflectors do not scale well and are able to support groups of tens of clients at most, their mesh is scalable enough to support a sufficient num-

ber of clients. Interesting direction of reflector development is the implementation of self-organizing and automatic discovery capabilities stemming from ideas of peer-to-peer networking. We will consider either the pure, hybrid or super-peer modes to define the best model for the reflectors self organization. The reflectors will be able to create overlay networks that can sustain even partial network disintegration without completely breaking overlay network.

As the reflectors can create the overlay networks, the reflector based solution does not depend on any specific functionality of the underlying network. All the advanced features are provided by higher user-controlled layers. Any group needing to collaborate can start its own reflector(s) and a unicast connectivity is the only required network capability from any client to the reflector. While the data routing and the data replication are automatically provided, user-specific services can be added as extensions (modules) to the reflector.

The future work will include control through grid service interfaces, as specified by an Open Grid Services Architecture framework [13]. This will enable easy integration with both next-generation collaborative Grid environments (e. g. AccessGrid version 2.x [14]) and with an optical network control plane, purposely built using the web/grid service approach. Thus the reflector will be able to cooperate easily with either the underlying network or other collaborative environment frameworks. The next direction in future work is concerned in development of user administration of reflector's data processing capabilities. Possible example of this administration is moderating data streams and creating communication environment for sub-group discussion inside the videoconferencing groups and fully moderated discussion like teaching in virtual classroom.

## 6 Acknowledgements

This research has been kindly supported by a research project "High Speed Research Network and its New Applications" (MŠM000000001) and "Optical Network of National Research and Its New Applications" (MŠM 6383917201). The authors would like to thank to Tomáš Rebok for helping with the implementation of performance evaluation tools.

## References

1. Chin Jr., G., Myers, J., Hoyt, D.: Social networks in the virtual science laboratory. Communications of the ACM **45** (2002)
2. Psounis, K.: Active networks: Applications, security, safety and architectures. IEEE Communication Surveys (1999)
3. Andersen, D., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient overlay networks. In: 18th ACM Symp. on Operating Systems Princpiles (SOSP), Banff, Canada (2001)
4. Denemark, J., Holub, P., Hladká, E.: RAP – Reflector Administration Protocol. Technical Report 9/2003, CESNET (2003)

5. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany (2001) 329–350
6. Rebok, T., Holub, P.: Synchronizing RTP Packet Reflector. Technical Report 7/2003, CESNET (2003)
7. Bouček, T.: Kryptografické zabezpečení videokonferencí. Master's thesis, Military Academy Brno (2002) Czech only.
8. Salvet, Z.: Enhanced UDP packet reflector for unfriendly environments. Technical Report 16/2001, CESNET (2001)
9. Internation Electrotechnical Commission: IEC 61834: Recording – Helical-scan digital video cassette recording system using 6,35 mm magnetic tape for consumer use (525-60, 625-50, 1125-60 and 1250-50 systems). (1998, 1999, 2001) Parts 1–10, `http://www.iec.ch`.
10. Ogawa, A., Kobayashi, K., Sugiura, K., Nakamura, O., Murai, J.: Design and implementation of DV based video over RTP. In: Packet Video Workshop 200. (2000) `http://www.sfc.wide.ad.jp/DVTS/pv2000/index.html`.
11. Hladká, E., Holub, P., Denemark, J.: Teleconferencing support for small groups. In: Terena Networking Conference '02, TERENA (2002) `http://www.terena.nl/tnc2002/proceedings.html`.
12. Hladká, E., Holub, P., Liška, M.: Modular communication reflector with dv transmission. In: VRS'04, PASNET (2004) Czech only.
13. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum (2002)
14. Childers, L., Disz, T., Olson, R., Papka, M.E., Stevens, R., Udeshi, T.: Access grid: Immersive group-to-group collaborative visualization. In: Proceedings of Immersive Projection Technology, Ames, Iowa (2000)

# Appendix D

# Communication Support as the User Tool

by Eva Hladká and Jiří Denemark

In 4th International Conference on Emerging e-learning Technologies and Applications, ICETA'2005. Košice, Slovakia, September 2005. Proceedings. elfa s. r. o., 2005. pp. 283–288, 6 p. ISBN 80-8086-016-6.

# Communication Support as the User Tool

Eva Hladká
*Faculty of Informatics*
*Masaryk University Brno*
*Czech Republic*
*eva@fi.muni.cz*

Jiří Denemark
*Faculty of Informatics*
*Masaryk University Brno*
*Czech Republic*
*jirka@ics.muni.cz*

D

## Abstract

*Besides the network support, collaborative environments generally comprise of an administration portal and user tools. User empowered replication units called active elements provide the last building block for the fully user controlled collaborative environment, adding user control over the necessary network properties. We developed a user empowered packet reflector whose directly controlled features are described here. Users may control the collaborative environment by means of setting up the reflector through a web interface and controlling does not depend on other videoconferencing tools. The major high-level features supported are recording, moderating, and subgrouping.*

## 1. Introduction

Various systems and technologies for videoconferencing and more generally for collaborative environment have been developed in the last 20 years. Common feature of these technologies was a separation in two parts: user tools controlled by end users and network support for multipoint data distribution. Multipoint data distribution can be based on native multicast (parallel communication schema), on a hardware or software replicating and distributing element called—reflector—(serial distribution schema), or on a combination of both approaches. While multicast is the most efficient approach which scales very well, its reliability and robustness are not sufficient. The reflector solution is independent of advanced network services but does not scale easily. Typical examples of videoconferencing collaborative systems are:

- Mbone,
- H.323,
- ISABEL.

In case of Mbone videoconferencing systems (including various types of videoconferencing systems like AccessGrid Point [1] and VRVS [2]), Mbone end-user application programs are used with a combination of native multicast, unicast bridges to multicast cloud on places where multicast is not available or on software reflectors. H.323 videoconferencing systems (MS Netmeeting, hardware clients) are using hardware or recently also software MCU (Multipoint Connection Unit). MCU does not scale well, for large videoconferences it is necessary to create a network of MCUs. ISABEL [3] creates its own data distribution infrastructure of reflectors and this infrastructure is able to optimize data flows during the session.

Native multicast is an advanced network service and is fully independent of users. Even network administrators have only limited chance to change the otherwise automatic multicast behavior. Network bridges to multicast cloud are in users hands but their role is only bridging to multicast-enabled part of the network without any user controlled features. Hardware MCU is from the user's point of view similar to multicast being part of the network infrastructure. The software reflector is the most flexible from this point of view.

After several years long experience with various videoconferencing systems, we learnt that creating an infrastructure for group communication fully controlled by users and overlaying the basic network is the necessary next step in development of communication infrastructure.

Let us discuss reasons for creating an overlay network infrastructure. The network is heterogeneous in many parameters and demands on network infrastructure of communicating groups are heterogeneous, too. General solutions pose great demands on network infrastructure and usually can satisfy only part of users' demands. If, instead of general solution for everybody, we could create a tailored infrastructure for each case, covering specific users' demands, this plethora of overlay networks limited

both in time and span could cover user demands in a much better way.

During the last years, the architecture of an active element fully controlled by the user has been created, implemented and used [4, 5, 6]. The goal of this paper is to show this element not only as a tool for creating overlay network for multipoint data distribution but in the role of user application tool that allows direct data transfer control during the videoconferences. This is the basic requirement to implement an environment that reflects situation encountered in direct human communication.

## 2. Active Element

Active element is a multiplication data unit. Its architecture is based on active router architecture described in [4] which has been modified to work entirely within the user space. Active element functionality is divided into two parts:

- data routing and processing,
- administration of the active element.

Data routing and processing is in detail described in [5, 6] and is not discussed in this paper. The second part—AE administration—is the main part for user to be able to use the active element directly as a user tool.

### 2.1. Administrative part of the AE

Communication with the reflector from the administrative point of view is provided using *messaging interfaces*, *management module*, and *administrative AAA module* of the reflector.

Messaging interface is a generic entity which can be instantiated as RPC, SOAP, HTTP interface with SSL/TLS support, for example, or as a simple TCP connection limited to the loopback interface of the machine running the reflector. Each of these interfaces unwraps the message if necessary and passes it to the management module.

The administrative protocol called RAP (Reflector Administration Protocol, [7]) has been defined for administrative communication with reflector modules using messaging interfaces.

An individual RAP parser is run for each administrative session. The parser uses callback functions registered by the module the parser is run from to read input data from the messaging interface. The parser then performs a request syntax validation and hands over each valid request to the management module.

Messaging interface architecture and its connections to the management module are illustrated in Figure 1.

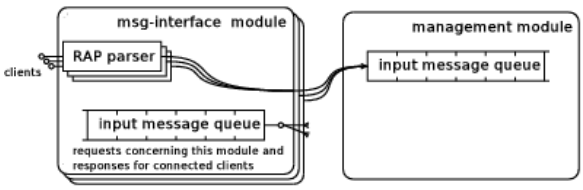



**Figure 1. Message interface schema**

The management module checks the validity of the message and its authenticity and authorization status, querying the administrative AAA module, which is also responsible for the processing of accounting information of accepted messages. The availability of sufficient resources to process accepted messages is checked with the resource management module and afterwards the message is passed to the appropriate module(s). The completion status is returned back to the management module, which notifies the administrative AAA module to close the accounting. If a failure occurs, its description is stored via the administrative AAA module and is sent back to the client via the client's messaging interface.

The same mechanism can be used for *logging* purposes. A command to invoke sending logging information can be issued using the messaging interface. One or more messaging interfaces may also be opened with the `LOGGING` flag set which causes the interface to request the reflector to send all logging information automatically. Messaging interfaces such as the reflector `syslog` interface can be used—it logs information without user intervention. Messaging interfaces receive events created within the reflector via the management module.

### 2.2. AE administration protocol

The protocol RAP is designed as soft-state, i.e., the connection is closed after a certain period of client inactivity. Keep-alive messages have to be sent if the client wants to maintain connection and does not have any requests to send.

The protocol uses request-response schema: each request is followed by one or more responses. If some messaging interface has `LOGGING` flag set it will send a request to obtain logging information according to the reflector startup configuration (e.g., such messaging interface can provide `syslog` interface).

The protocol also contains a module identification to be able to deliver requests to selected modules only.

RAP is designed to be easily extensible without the need for changing its specification. Any reflector's module can specify its own RAP requests controlling specific behaviour of a particular module. By sending requests to modules providing a special functionality, a

user is able to control all additional features of the active element.

## 3. Active Element Features

The basic information about the AE we described is sufficient to demonstrate that various functions can be designed and implemented. In unity with the focus of this paper, three functions which transform the AE into the collaborative user tool will be described in detail.

### 3.1. Recording

Some videoconferencing tools support recording, so that the data sent/received can be stored locally (by a client) for later inspection or replay. However, not all tools support this activity, and sometimes it may be beneficial to have a single storage point that records all the data transmitted during a session. The centralized data recording facility has many advantages over much simpler features of most videoconferencing clients. It may be a trusted neutral agent (no local editing of content), it can guarantee to store all data actually transmitted (in contrast to a local client which may experience some local data loss) and it may be more efficient (just one copy of the data is created).

The recording module within the reflector is controlled by end users and it may be used to recording or replaying the whole communication independently of users' tools. The fact that data generated by all tools used for a communication are stored in a single place allows a synchronized replay of all streams.

### 3.2. Moderating

In parallel collaboration within a group of people, group members perform activities in parallel while trying to coordinate these activities at the same time. This type of collaboration is usually quite difficult and complex, as people must coordinate their actions on the same subject. It is the interaction among group members that determines the outcome of a particular collaboration. A whole area of psychological science—group dynamics—is dedicated to this problem [8].

With an increased number of group members weighted interactivity of all members decreases and at some point efficiency of collaboration can be increased only by appointing somebody to administer the communication. Also the multi-directionality of communication among members decreases and from some point a one-way communication prevails.

For example, two persons are working on the same text. They are sitting opposite each other and speaking about the subject. Their communication is bidirectional and equal. When a third person joins the group, communication triangle is created and the work continues with three people in a similar pattern. It will be the same case with four, five, six, and seven persons, even when this is accompanied with the growing problems associated with multi-directional communication. Increasing complexity of such a communication is illustrated in Figure 2.



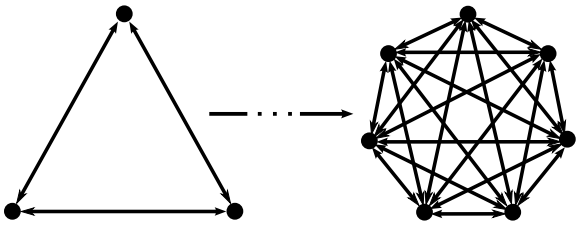


**Figure 2. Multi-directional communication schema**

However, for eight or more persons in a non-moderated group (i.e. all members are equal and have the same position within a group), the efficiency of communication is dramatically reduced. So with a growing number of participants, one (or very small number) of them must become moderators of the communication and one-way communication begins to dominate when one person speaks and others mostly listen. It becomes the moderator's decision who will speak next. A communication model with a moderator is shown in Figure 3.

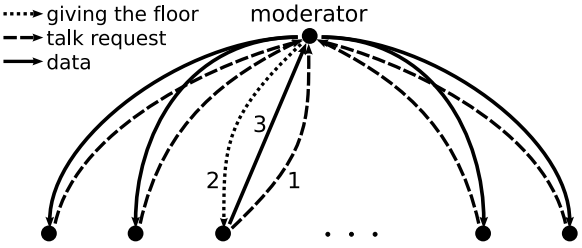



**Figure 3. Moderated communication**

For non-virtual moderated collaboration environment when all persons are present at the same place, the efficiency of communication relies on all listening people being quiet and not disturbing. On the other hand, collaboration in a distributed group of people connected with network lines has an important advantage—the moderator can enforce her/his decision by cutting data from all persons except for the one who was given the floor. In this section, we will focus on a communication support for moderated communication within a

distributed groups and how a user (as a moderator) can control the communication layer.

The possibility to give the floor and cut off other streams is important mainly for large groups of people with members who are not involved in the communication enough to keep attention and be quiet. Such a situation happens e.g. in education—not all students are interested in all lectures and they could disturb others if the teacher (moderator) did not have an option to restrict data transfer.

Also groups which require high quality communication accompanied by high transmission bandwidth utilization may benefit from moderating as group members (except for moderator) have to receive only a limited number of data streams. Besides, the number of streams is constant and independent of a number of group members. Still, the network line from moderator to a reflector may easily become a bottleneck of a whole environment. To eliminate such constriction a transcoding feature [6] of the reflector can be configured leaving only the selected stream intact and transcoding all other streams (mostly used for floor requests) into a lower quality. Thus, each participant (including the moderator) receives only one full-quality stream.

The moderating feature is provided by a special processing module of the reflector. According to moderator's requests, the module decides which RTP and RTCP packets may pass through the reflector to a particular person without being changed, which of them have to be changed or replaced, and which packets are discarded. Headers of RTP and RTCP packets sent by a person who was given the floor are altered in such a way that the packets seem to be sent by an additional virtual member of a communicating group. Thus, all data packets from people sequentially chosen by a moderator to talk form a single data stream regardless of a particular speaker.

By now, speakers are distinguished by their IP address which prevents distributed reflectors to be used for better scalability. In the near future, the module will be extended to use SSRC field of RTP header and canonical name of a source (CNAME item of RTPC packets) [rfc3550] as a unique identifier of each participant independently of a machine from which a reflector receives data packets.

### 3.3. Subgrouping

In common life when people meet to cooperate, they often (and especially in large groups of people) need to discuss some details of a topic within an isolated group (subgroup) so that other participants cannot overhear such discussion. We use either whispering or move to another place. Both of these solutions may have their disadvantages—persons which are physically close to the whispering group are still able to overhear at least some parts of the communication while the second solution requires people to actually move and once they are moved, they cannot hear the rest of the group any more.

Such functionality have to be supported directly by a communication layer otherwise the only way of satisfying the need for creating isolated subgroup is to create stand-alone group which is the equivalent of changing place. Besides, participants who want to create a subgroup have to be physically isolated. By adding support for subgrouping into a communication layer, all disadvantages of non-distributed collaboration environment may be bypassed by separating a subgroup from the rest of a whole group by a "semi-permeable wall". Thus, private communication among members of a subgroup cannot be overheard by other people while a discussion outside the subgroup is still distributed to all (including subgroup members) participants. Communication schema within a group with private subgroup is illustrated in Figure 4.

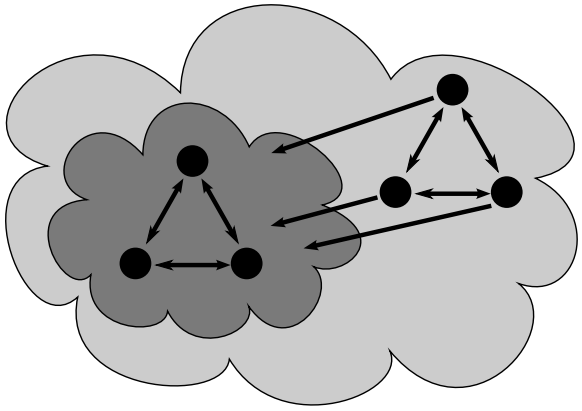


**Figure 4. Subgrouping**

Implementation of subgrouping for our reflector-based communication architecture is quite straightforward as it does not require any data processing. The module designed to support subgrouping just alters a list of clients each packet is to be send to. A decision which clients should be removed from distribution list is based on user request specifying all members of a subgroup. Current design of the module uses IP addresses as users' identifier and it will be changed in the same way as moderating module for distributed reflector architecture.

## 4. User Interface

To provide a user interface, we associated the web server with the reflector. This association allows us to

create an end-user interface as well as an administrator interface for the reflector without introducing any new unexpected functionality for the end user to learn. Also, the availability of web browsers for practically all computer and operating systems platforms contributed to our choice of this technology to open higher level access to the reflector and its functionality.

The web portal that is currently being developed is composed of two major parts—informational and administrative. The informational part is publicly accessible and it is used to advertise information about the reflector and current as well as planned conferences. Access to the administrative part is authenticated by login name and a password and uses SSL for securing the communication.

The administrative part serves to manage conferences and reflectors (although each user does not need to use more than one reflector) allowing a user to start and stop her/his reflector as well as perform such complex tasks as moderating the conference or creating a private subgroup.

The web interface can be used to control both local and remote reflectors. Local reflector is started directly through the web interface and is running on the same host. A user can set up one local reflector at most. Obviously, remote reflector is running on any host and it is user's responsibility to start the reflector and set it up so that the web interface can communicate with it. In contrast to local reflector, each user can control any number of remote reflectors. Thus, even reflectors running on host with no web server can still be controlled comfortably using the web interface.

## 5. Videoconferencing with AE

The processing and distribution of UDP packets which is at the core of the active element provides support for variety of end-user collaborative applications. We use the active element based reflectors routinely with client tools like `vic`, `rat`, and `wbd` from Mbone tools package. The user interface of the reflector gives direct control over the data processing to the end user's hands.

As current high-quality videoconferencing tools tend to utilize high-quality and high-bandwidth multimedia streams, we have also successfully tested the reflector with the DV over IP transmission tools from the DVTS project [11].

## 6. Conclusions

We decided to provide group data distribution needed for collaborative environments through the serial communication schema. This schema is enabled by the technology of active networks. The active element based on active nodes fully under end users' control allow to design and implement new functions that are used and controlled by users during the actual data transmission. Three such functions

- recording,
- moderating,
- subgrouping

are described in details. The actual reflector control is made available through the web portal, which also support planning the collaborative events and play the controlling and informational role. The active element became a new type of user tool with functions under direct users' control. This setup empowers users with the capability to create the synchronous communication environments as realistic as possible.

These three features were inspired by users' demands and were created for concrete users' groups. Of course it is possible to create other functions and modular architecture of reflector is flexible enough to allow for their easy implementation.

It is evident that active element does not scale well. In related paper this problem was studied and a solution provided [10]. Thanks to active element's modular architecture it is possible to create overlay networks from AE with no worse scalability and more robust then multicast solution. However, having more than one AE, we added possibility to change the AEs network dynamically during the collaborative session. Taking into account that each AE has its administrator, it leads to a necessity to define correct form of user-controlled functions to make them able to cooperate. This is one of the future goals of development of AE.

## 7. Acknowledgement

This research is supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201). We would also like to thank to Assoc. Prof. Luděk Matyska for stimulating discussions on virtual human communication and for proof reading this text.

## 8. References

[1] AccessGrid. `http://www.accessgrid.org/`

[2] Virtual Rooms VideoConferencing System (VRVS). `http://www.vrvs.org/`

[3] ISABEL. `http://isabel.dit.upm.es/`

[4] E. Hladká, Z. Salvet. *An Active Network Architecture: Distributed Computer or Transport Medium*. Heidelberg: Springer-Verlag Berlin, 2001. LNCS 2094.

[5] E. Hladká, P. Holub, and J. Denemark. *User Empowered Virtual Multicast for Multimedia Communication*. In ICN'2004 Conference Proceedings. March 2004.

[6] E. Hladká, P. Holub, and J. Denemark. *User-Empowered Programmable Network Support for Collaborative*

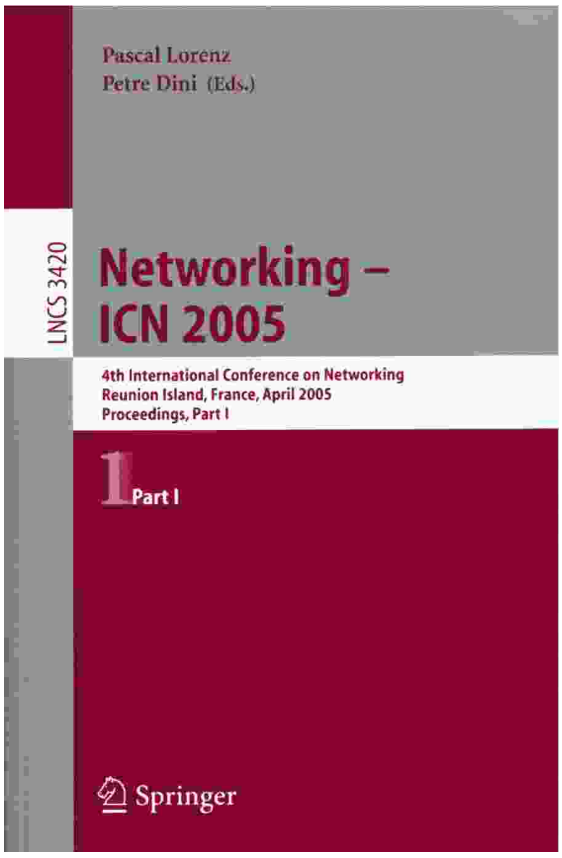
*Environment*. In Universal Multiservice Networks: Third European Conference, ECUMN 2004. Springer-Verlag Heidelberg. October 2004. Pages 367–376.
[7] J. Denemark, P. Holub, and E. Hladká. *RAP—Reflector Administration Protocol*. Technical report. CESNET z. s. p. o., 2003. 32 pages. 9/2003.
[8] D. R. Forsyth. *Group Dynamics*. Wadsworth Publishing Company, 3rd edition, 1999.
[9] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. RFC 3550. June 2003.
[10] P. Holub, E. Hladká, and L. Matyska. *Scalability and Robustness of Virtual Multicast for Synchronous Multimedia Distribution*. In Networking—ICN 2005: 4th International Conference on Networking
[11] E. Hladká, M. Liška, and T. Rebok. Stereo video over IP network. Proceedings of International Conference on Networking and Services 2005. Accepted paper.

# Appendix E

# Scalability and Robustness of Virtual Multicast for Synchronous Multimedia Distribution

by Petr Holub, Eva Hladká and Luděk Matyska

In 4th International Conference on Networking, ICN 2005, Reunion Island, France, April 2005. Proceedings. Lecture Notes in Computer Science 3421, Heidelberg: Springer Berlin, 2005. 8 p. ISSN 0302-9743.



# Scalability and Robustness of Virtual Multicast for Synchronous Multimedia Distribution

Petr Holub[1,2], Eva Hladká[1], and Luděk Matyska[1,2]

[1] Faculty of Informatics
[2] Institute of Computer Science,
Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
hopet@ics.muni.cz, eva@fi.muni.cz, ludek@ics.muni.cz

**Abstract.** A simple UDP packet reflector for virtual multicast multimedia transfer is extended to form a distributed system of active elements that solves the scalability problem of otherwise centralistic approach. The robustness of such virtual multicast delivery system is also discussed and shown to be better than the native multicast can offer. The maximum latency, important for multimedia transfer and related to the number of hops through the network of active elements, can be kept bounded. Possible support for synchronized multi-stream transfer is also discussed.

## 1 Introduction

A virtual multicasting environment, based on an active network element called "reflector" [1] has been successfully used for user-empowered synchronous multimedia distribution across wide area networks. While quite robust replacement for native, but not reliable multicast used for videoconferencing and virtual collaborative environment for small groups, its wider deployment is limited by scalability issues. This is especially important when high-bandwidth multimedia formats like Digital Video are used, when processing and/or network capacity of the reflector can easily be saturated.

A simple network of reflectors [2] is a robust solution minimizing additional latency (number of hops within the network), but it still has rather limited scalability. In this paper, we study scalable and robust synchronous multimedia distribution approaches with more efficient application-level distribution schemes. The latency induced by the network is one of the most important parameters, as the primary use is for the real-time collaborative environments. We use the overlay network approach, where active elements operate on an application level orthogonal to the basic network infrastructure. This approach supports stability through components isolation, reducing complex and often unpredictable interactions of components across network layers.

## 2 Synchronous Multimedia Distribution Networks

Real-time virtual collaboration needs a synchronous multimedia distribution network that operates at high capacity and low latency. Such a network can be

composed of interconnected service elements—so called *active elements* (AEs). They are a generalization of the user-empowered programmable reflector [1].

The reflector is a programmable network element that replicates and optionally processes incoming data usually in the form of UDP datagrams, using unicast communication only. If the data is sent to all the listening clients, the number of data copies is equal to the number of the clients, and the limiting outbound traffic grows with $n(n-1)$, where $n$ is the number of sending clients. The reflector has been designed and implemented as a user-controlled modular programmable router, which can optionally be linked with special processing modules in run-time. It runs entirely in user-space and thus it works without need for administrative privileges on the host computer.

The AEs add networking capability, i. e. inter-element communication, and also capability to distribute its modules over a tightly coupled cluster. Only the networking capability is important for scalable environments discussed in this paper.

Local service disruption—element outages or link breaks—are common events in large distributed systems like wide area networks and the maximum robustness needs to be naturally incorporated into the design of the synchronous distribution networks. While the maximum robustness is needed for network organization based on out-of-band control messages, in our case based on user empowered peer to peer networks (P2P) approach described in Sections 3.1 and 5, the actual content distribution needs carefully balanced solution between robustness and performance as discussed in Section 4. The content distribution models are based on the idea that even sophisticated, redundant, and computationally demanding approaches can be employed for smaller groups (of users, links, network elements, ...), as opposed to simpler algorithms necessary for large distributed systems (such as the global Internet). A specialized routing algorithm based on similar ideas has been shown, e. g. as part of the RON approach [3].

## 3 Active Element with Network Management Capabilities

As already mentioned in Sec. 2, the AE is the extended reflector with the capability to create network of active elements to deploy scalable distribution scenarios. The network management is implemented via two modules dynamically linked to the AE in the run-time: Network Management (NM) and Network Information Service (NIS). The NM takes care of building and managing the network of AEs, joining new content groups and leaving old ones, and reorganizing the network in case of link failure.

The NIS serves multiple purposes. It gathers and publishes information about the specific AE (e. g. available network and processing capacity), about the network of AEs, about properties important for synchronous multimedia distribution (e. g. pairwise one-way delay, RTT, estimated link capacity). Further, it takes care of information on content and available formats distributed by the

network. It can also provide information about special capabilities of the specific AE, such as multimedia transcoding capability.

The NM and NIS modules can communicate with the AE administrator using administrative modules of the AE kernel. This provides authentication, authorization, and accounting features built into the AE anyway and it can also use Reflector Administration Protocol (RAP) [4] enriched by commands specific for NM and NIS. The NM communicates with the Session Management module in the AE kernel to modify packet distribution lists according to participation of the AE in selected content/format groups.



### 3.1 Organization of AE Networks

For the out-of-band control messages, the AE network uses self-organizing principles already successfully implemented in common peer to peer network frameworks [5],[6], namely for AE discovery, available services and content discovery, topology maintenance, and also for control channel management. The P2P approach satisfies requirements on both robustness and user-empowered approach and its lower efficiency has no significant impact as it routes administrative data only.

The AE discovery procedure provides capability to find other AEs to create or join the network. The static discovery relies on a set of predefined IP addresses of other AEs, while the dynamic discovery uses either broadcasting or multicasting capabilities of underlying networks to discover AE neighborhood. Topology maintenance (especially broadcast of link state information), exchange of information from NIS modules, content distribution group joins and keep-alives, client migration requests, and other similar services also use the P2P message passing operations of AEs.

### 3.2 Re-balancing and Fail-Over Operations

The topology and use pattern of any network changes rather frequently, and these changes must be reflected in the overlay network, too. We consider two basic scenarios: (1) re-balancing is scheduled due to either use pattern change or introduction of new links and/or nodes, i. e. there is no link or AE failure, and (2) a reaction to a sudden failure.

In the first scenario, the infrastructure re-balances to a new topology and then switches to sending data over it. Since it is possible to send data simultaneously over both old and new topology for very short period of time (what might result in short term infrastructure overloading) and either the last reflector on the path or the application itself discards the duplicate data, clients observe seamless migration and are subject to no delay and/or packet loss due to the topology switch. This scenario also applies when a client migrates to other reflector because of insufficient perceived quality of data stream.

On the contrary, a sudden failure in the second scenario is likely to result in packet loss (for unreliable transmission like UDP) or delay (for reliable protocols like TCP), unless the network distribution model has some permanent

redundancy built in. While multicast doesn't have such a permanent redundancy property, the client perceives loss/delay until a new route between the source and the client is found. Also in the overlay network of AE without permanent redundancy, the client needs to discover and connect to new AE. This process can be sped up when client uses cached data about other AEs (from the initial discovery or as a result of regular updated of the topology). For some applications, this approach may not be sufficiently fast and permanent redundancy must be applied: the client is continuously connected to at least two AEs and discards the redundant data. When one AE fails, the client immediately tries to restore the degree of redundancy by connecting to another AE. The same redundancy model is employed for data distribution inside the network of AEs, so that re-balancing has no adverse effect on the connected clients.

The probability of failure of a particular link or AE is rather small, despite high frequence of failures in global view of large networks. Thus the two fold redundancy ($k = 2$) might be sufficient for majority of applications, with possibility to increase ($k > 2$) for the most demanding applications.

## 4 Distribution Models

### 4.1 Multicast Schemes

In an ideal case, the multicast organization of the data distribution is the most efficient scheme to distribute data to multiple clients. However, it is very difficult for a user to place AEs into the physical network topology in such a way that no data will pass through any physical link twice. The only exception may be when AE network is implemented as a network of active routers, but this goes against the user-empowered approach we support. Thus the multicast paradigm is only an upper-limit on efficiency of the distribution.

There are two basic approaches to build multicast distribution tree: source-based tree also known as shortest path tree (SPT) and shared tree. Regarding the synchronous character of multimedia data distribution, the SPT with reverse path forwarding (RPT) has two major advantages: it minimizes latency compared to shared tree where the data is sent through rendezvous point and it provides shortest paths between the source and the receivers (advantage for large volume of data transmission).

To build SPTs, it is necessary to have underlying unicast routing information. This information can be maintained very efficiently by RON [3]. As an addition to fast convergence in case of network link failure, it is possible to define policy to select the shortest path not based on hop count, but based on path round trip time or even one way propagation delay if available.

**Fail-Over Operation** Standard operation when the link failure occurs is to build a new SPT as described above. If even the convergence speed of RON is not acceptable, there is another possible strategy to minimize delay due to SPT reconstruction. It is possible to compute multiple SPTs at the same time,

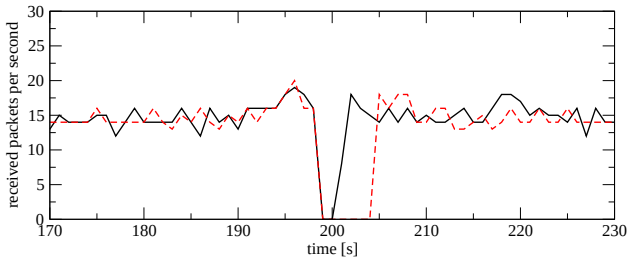



**Fig. 1.** Recovery time for with backup SPT (solid line) and without it (dashed line) simulated using cnet-based network simulator.

choose single SPT for data distribution and keep the remaining SPTs for fail-over operation. For permanent redundancy scenario, more than one SPT can be used simultaneously and duplicate data will be discarded by client applications. In full graph, there are $n^2 - n$ links between the AEs. For a small number of AEs, alternative SPTs can be computed that don't use one selected link at a time. If that particular link fails, the alternative SPT can be immediately switched on. For larger number of AEs where number of links is too large, it is possible to compute $\lfloor n/2 \rfloor$ possible SPTs with disjunct set of links. When using SPTs or shared trees (ST) with backup based on disjunct sets of links, it is necessary to ensure that not all links from one AE are used in one SPT/ST, since the AE would become isolated in backup SPT/ST. When backup SPT/ST is available, the network recovery is limited just by broadcast speed to announce switching to a new SPT/ST, but when there is no backup, the alternative SPT/ST must be computed first (Fig. 1). During the normal operation, all these SPTs are monitored for their usability and when link fails in the current SPT, the original SPT can be swapped for another working SPT if at least one other usable SPT is available.

### 4.2 2D Full Mesh

The simplest model with higher redundance, serving also as the worst case estimate in terms of scalability, is a complete graph in which each AE communicates directly with all the remaining AEs. This *2D full-mesh tunneling* model was studied and described in detail in [2]

Let's assume a 2D full mesh of reflectors, each populated with $n_r$ clients. The limiting traffic in this mesh is again the outbound traffic on the AE which scales as $out = n_r^2 m + n_r(m-2)$.

**Fail-Over Operation** When a link or whole AE drops out in the full mesh, the accident only influences data distribution from/to the clients connected to that AE. In case of link failure inside the AE mesh, the client is requested to

migrate to an alternative AE. In case that AE itself fails the client initiates migration on its own. Alternative AEs should be selected randomly to distribute load increase more evenly and the load increase will be $\lceil \frac{n_r}{m-1} \rceil$. When even this migration delay is not acceptable, it is possible for a client to be permanently connected to an alternative AE and just switch the communication. For even more demanding applications, the client can use more than one AE for sending in parallel.

Although this model seems to be fairly trivial and not that interesting, it has two basic advantages: first, the model is robust and failure of one node influences only data from/to the clients connected to that AE. Second, it introduces only minimal latency because the data flows over two AEs at most. Next we will examine another model that has the same latency and robustness properties but that scales better.



### 4.3 3D Layered-Mesh Network

The layered mesh model creates $k$ layers, in which data from a single AE are only distributed. That means each client is connected to one layer for both sending and receiving (sending only if $n_r = 1$; in other cases the client needs to receive data from remaining $n_r - 1$ clients of the AE used for sending) and to all other layers for receiving only. Each layer comprises 2D full mesh of $m$ AEs. For the sake of simplicity, we first assume that $k = m$ and each AE has $n_r$ clients, thus $n_r = \frac{n}{m} = \frac{n}{k}$.

In this scenario, the number of inbound streams is $in = n_r$. Number of outbound streams is $out_{s/r} = n_r^2 + n_r(m-2)$ if the sending client is connected to this particular AE, and $out_r = n_r^2$ when only receiving clients are connected.

This model is problematic because of increasing the number of AEs used. However it seems to be the last model that doesn't introduce intermediate hops and thus keeps hop-count at minimum.

**Intermediate AEs** Let's create $q$-nary tree used for distributing data from AE with sending clients to $m-1$ AEs with listening clients. When building $q$-nary tree with $\lambda$ intermediate layers $\lambda = \log_q(m-1) - 1$, the total number of intermediate AEs is $L = \sum_{p=1}^{\lambda} q^p = \frac{m-1-q}{q-1}$.

Flows in this type of network are as follows: $out_{s/r} = n_r(n_r - 1) + qn_r$ for outer AE with sending clients connected, $out_r = n_r^2$ for outer AE with only receiving clients, and $out_i = qn_r$ for inner intermediate AEs. For all types of AEs, the number of inbound flows is $n_r$.

There are however two disadvantages of this model:

- The number of hops inside the mesh of AEs increases by $\lambda$ compared to the plain 3D mesh model.
- Compared to the plain 3D model, the number of the intermediate AEs further increases to $m_{tot} = mk + Lk$. For $m = k$, it becomes $m_{tot} = m(m + L)$.

Nevertheless, this model provides the same redundancy while improving scalability compared to the simple 2D mesh.

**Fail-Over Operation** Each of the mesh layers monitors its connectivity. When some layer disintegrates and becomes discontinuous, the information is broadcasted throughout the layer and to its clients. The clients that used that layer for sending are requested to migrate to randomly chosen layer from the remaining $k-1$ layers and the listening-only clients simply disconnect from this layer. Such behavior increases load on the remaining $k-1$ layers but as the clients choose the new layer randomly, the load increases in roughly uniform way by $\lceil \frac{n_r}{k-1} \rceil$.



## 5 Content Organization

The multimedia content can be encoded in many different formats, that suit specific needs or capabilities of the network and the listening clients. In some cases (e. g. MPEG-4 formats) the highest quality format can be decomposed into $N$ different layers (groups) that are sent over network independently. When native multicast is used, the client subscribes for the first $M \in \langle 1; N \rangle$ groups only, thus controlling the quality reduction of received content. With native multicast, there is no easy way to prioritize and synchronize the streams, which may lead to unexpected loss of quality (if data in the first layer are lost, the other layers may render useless).

As AEs support also multimedia transcoding (capable of being *active gateways*), an extended approach can be used. The format decomposition or even transcoding to completely different format may be performed by an AE, providing a flexible on demand service–the transcoding occurs only if really needed by some client. Also, the AEs are capable of synchronizing individual streams—they "understand" the decomposition and may re-synchronize individual streams. In case of severe overload, the higher (less important) stream layers are dropped first (again, AEs know the hierarchy), so the transmission quality is minimally affected.

To formalize our approach, we have designed three layer hierarchy:

- *content groups*—the highest level, an aggregation of several contents; it can be for instance a videoconferencing group (e. g. video and audio streams of individual videoconference participants)
- *content*—intermediate level, a content (a video stream, format independent)
- *format*—the lowest level, format definition.

Each multimedia stream in the network is then characterized by `(content group, content, format)` triplet which creates one record in the SPT tree. The available formats for each content create an oriented graph where the root is the source format and the child nodes define the formats created from their parents. A client can choose the best suitable format, or different formats for different contents within one content group (e. g. a lecturer's stream with the highest quality).

The information about available content groups, content, and available formats is published via NIS on AEs and is distributed and shared across the network of AEs.

## 6 Related Work

There are a few known applications for synchronous distribution of multimedia data over IP networks. Probably the most important ones are cascading of H.323 multi-point connection units (MCUs) and Virtual Room Videoconferencing System (VRVS). The networks of H.323 MCUs are based on a static preconfigured topology and they don't offer user-empowered approach. The VRVS is only provided as a service and the users' traffic is managed by VRVS administrators. Also, although the VRVS team reports some move in favor of more elaborate and dynamic network of reflectors, we believe that creating flexible user-empowered multimedia network is more suited for open systems without centralized administration.



## 7 Conclusions

In this paper the models for the virtual multicast scalability were introduced with discussion of robustness and fail over capabilities of the proposed solutions. We have implemented a prototype of Active Element suitable for simple networking scenarios for Linux and FreeBSD operating systems and the models have also been verified using network simulator. The AE network organization support is being implemented based on JXTA 2.0 P2P framework. The full application-level multicast data distribution with multicast subgroups as described in Secs. 4.1 and 5 is under development.

## 8 Acknowledgment

This research is supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201). We would also like to thank to Tomáš Rebok for helping with implementation of network simulations.

## References

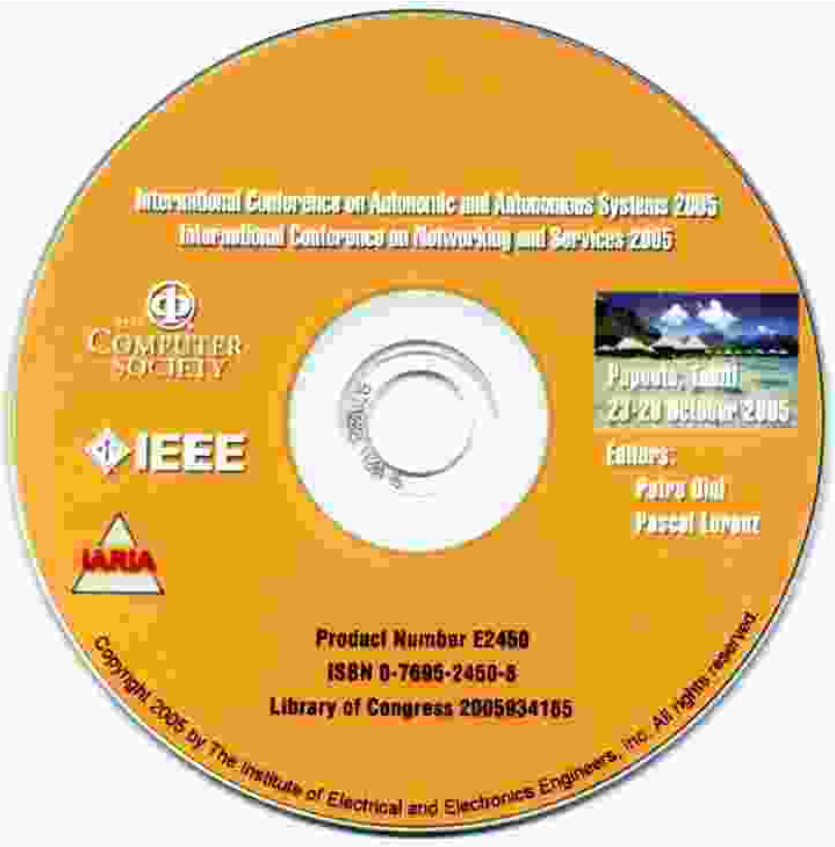
1. Hladká, E., Holub, P., Denemark, J.: User empowered virtual multicast for multimedia communication. In: Proceedings of ICN 2004. (2004)
2. Hladká, E., Holub, P., Denemark, J.: User empowered programmable network support for collaborative environment. In: ECUMN'04. Volume 3262/2004 of Lecture Notes in Computer Science., Springer-Verlag Heidelberg (2004) 367 – 376
3. Andersen, D., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient overlay networks. In: 18th ACM Symp. on Operating Systems Princpiles (SOSP), Banff, Canada (2001)
4. Denemark, J., Holub, P., Hladká, E.: RAP – Reflector Administration Protocol. Technical Report 9/2003, CESNET (2003)
5. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany (2001) 329–350
6. Yang, B., Garcia-Molina, H.: Designing a super-peer network. In: IEEE International Conference on Data Engineering. (2003) 25

# Appendix F

# Stereoscopic Video over IP Networks

by Eva Hladká, Miloš Liška and Tomáš Rebok

In The First International Conference on Systems and Networks Communications, ICNS 2005. Papeete, Tahiti, October 2005. Proceedings. Institute of Electrical and Electronics Engineers, 6 p. ISBN D-7695-2450-8.



# Stereoscopic Video over IP Networks

Eva Hladká[1,2], Miloš Liška[1], and Tomáš Rebok[1]
[1] Faculty of Informatics, Masaryk University, 602 00 Brno, Czech Republic
[2] CESNET z. s. p. o., Zikova 4, 160 00 Praha 6, Czech Republic
E-mail: {eva,xliska,xrebok}@fi.muni.cz

## Abstract

*Transfers of high-quality multimedia content pose new demands on capacity and services provided by the contemporary high-speed computer networks. Transfer of stereoscopic video is a specific example, as it needs synchronization between two separate data streams. We have set up a stereoscopic video capture system and studied synchronization of two separate Digital Video format streams sent over packet networks. We have adapted application tools to support the synchronization and used an active element working as a synchronizing UDP packet reflector to explicitly synchronize the streams if they are desynchronized in the network. We have experimentally studied the quality of achievable synchronization and the relationship between the amount of desynchronization and the additional latency overhead posed by buffering of the data on the synchronizing reflector. The results prove our assumption that even high-quality DV streams can be successfully synchronized using the simple packet reflector running on common IA32-based computer.*

## 1. Introduction

Multimedia transfers are becoming one of the most important applications for current high-speed computer networks. New services, that are being developed for high-performance multimedia processing and transport, are often used to create virtual collaborative environments, where people can interact regardless of geographical distance between them. Most of the contemporary collaborative environments work in 2D only, meaning that the reality, which has three dimensions (3D), is reduced to 2D picture with inevitable loss of some information connected with the depth of the space. To simulate 3D environment by stereoscopic image, two streams must be transmitted over the network synchronously, one for each eye. To provide natural perception, the quality of individual streams must be rather high and thus high-resolution image with low compression needs to be deployed resulting in high data rate. Such requirements are satisfied e. g. by Digital Video format. Transmission and synchronization of stereoscopic video streams in DV format over the high-speed network are studied in this paper.

Processing of multimedia content usually comprises three phases: acquisition, transport, and presentation. The acquisition of stereoscopic video capture uses two cameras that mimic two human eyes. The two resulting streams need to be transmitted over the network and received synchronously by the display system. However, common computer networks can't enforce directly this kind of synchronization. One possibility is to synchronize and multiplex data at the source and send both video streams in one data (packet) stream. However, processing both streams on a single machine might not be feasible depending on video format used for the transmission. Our solution is thus to synchronize otherwise independently transmitted streams at some point in the network close to the display nodes, where suitable active element is placed. For purpose of our evaluation, the general purpose active element has been implemented using simple synchronizing UDP packet reflector.

## 2. Digital Video

In order to create highly realistic and information-rich environment, we have opted for using Digital Video (DV) format as the basic video format for our experiments with stereoscopic video. It provides very good image quality (with limits given by PAL resolution) while having reasonable compression ratio (approx. 5:1), low latency compression and decompression process based on I-frames only, and sustainable requirements on processing infrastructure.

The DV video transmission over IP networks has been standardized and implemented by DVTS project [1] for several operating systems (e.g. Linux, FreeBSD, Windows 2000/XP, and MaxOS X). The DVTS project originally included (i) IEEE-1394 kernel

driver for FreeBSD, (ii) `dvsend` for sending the DV video captured from IEEE-1394 to the network as a RTP stream, (iii) `dvrecv` for receiving the DV stream from the network and saving it locally to the disk, and (iv) `dvplay` for sending any DV data to the IEEE-1394 device. The tools produced by the DVTS project are stable and mature with the exception of the `xdvshow` prototype tool for displaying video locally using X-Windows interface. We have reimplemented the `xdvshow` tool, as its functionality is crucial for high quality display of stereoscopic video.

The original `xdvshow` implementation used just one thread for all operations on the DV video. The mutual exclusivity of the reading, decoding and displaying of one video frame was handled by busy waiting. The CPU consumption was unacceptably high, while CPU was spending most of the time in the busy waiting loop. The stereoscopic video display needs two DV streams synchronously and that means it should be possible two running instances of `xdvshow` at least on a single high-end computer.

The multi-threaded implementation solves the busy waiting problem sparing CPU time—this may be up to 50% of the total CPU time on modern CPUs. The multi-threaded architecture also allows separation of the reading process from decoding and displaying of the DV video. The multi-threaded `xdvshow` implementation uses three primary threads: One thread reads the DV stream from selected input, the second one is used for decoding of the stream and displaying the decoded video data and the third one is used for decoding and playing the audio part of the DV stream. There may be an additional thread used for interacting with the packet reflector, when it is the source of the DV stream. System of semaphores is used to thread cooperation and mutual exclusion for shared buffers' access. The `xdvshow` uses two shared buffers, one for DV video frames and the other for corresponding audio frames.

## 3. DV over IP

RTP protocol [2] is the major real-time transmission protocol used for multimedia distribution in IP networks. RTP is non-reliable and non-guaranteed service over the underlying UDP protocol. Also, QoS parameters of the transmission are not guaranteed by the protocol and must be handled on the application level if needed. The data transport by the RTP protocol is complemented by the RTCP protocol that provides support for delivery monitoring and also for control and identification functionality.

Standardization of DV over IP transmission using RTP protocol has been introduced by K. Kobayashi *et al.* in [3, 4]. A video frame in the DV format is divided into several "DIF sequences." A DIF sequence is composed of 80-bytes long DIF blocks. A DIF block is a primitive and atomic unit for all operations over the DV stream. Each RTP packet starts with an RTP header and no additional header is required for DV over IP transmission. The atomic DIF blocks are placed directly after the RTP packet header. It is possible to place any number of DIF blocks representing one distinct frame into one packet. The DIF blocks belonging to the next frame must be transmitted in a new RTP packet to facilitate frame detection. Since the RTP payload contains an integral number of DIF blocks, the length of RTP payload is divisible by 80.

The DV video transmission extensively uses Timestamp and Marker bit arrays in the RTP header. The time when the first data in a particular frame has been sent is stored in the timestamp array. All RTP packets in one videoframe must have the same timestamp according to the above mentioned standards. The timestamp increment for 25 fps PAL and 29.97 fps NTSC video are 3600 and 3003 respectively. The marker bit, left for any user-defined data by the RTP standard, is used to mark the last packet of the frame. When such packet is received, the whole frame can be immediately displayed, instead of waiting for the next packet to recognize the end of the frame. This mechanism reduces the total latency. However, the detection of end of frame must not rely on the marker bit presence only as the corresponding packet may be lost, and the check on the timestamp change in the RTP header must be always performed, too.

It is possible to transport the audio and video data in the same stream or separately. The choice must be done once and forever for one stream. It is also reflected in the dynamic Payload type and thus it must not change until the end of the RTP session.

The DV format video uses bandwidth of 25 Mbps. When audio data and header overhead is added, the resulting stream uses approximately 30 Mbps per each stream.

## 4. Stereoscopic Video

Nowadays, most of the video material transmitted over the network depicts scenes in two dimensions only. While human eye or better human brain is able to recognize quite lot of depth cues even in two dimensional picture to create some idea of space, realistic perception can not be achieved without taking the human eye anatomy into accounts. In real-world conditions, each eye perceives independently under a little bit different angle and the brain superimposes both images to get full

3D perception. This means we need to capture two independent video images for each eye to provide perception of the third dimension. The schematics of such a 3D (stereo) video capture system is depicted in Fig. 1. The theoretical model of such system is given in [5] and [6].

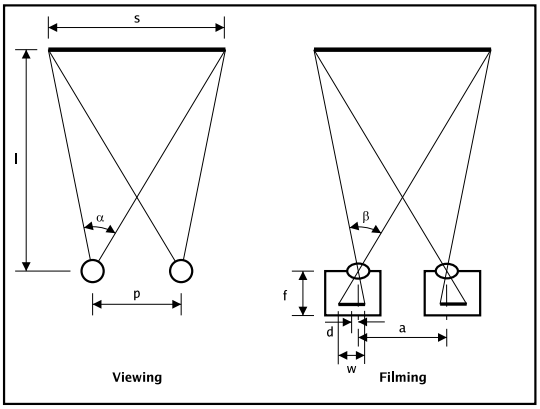



Figure 1. Setting up the cameras

To fulfill at least some of the preconditions, like the distance of the camera lenses focal points should be the same as the distance between human eyes focal points, a suitable camera mount needs to be used. After experimenting with a simple solution suggested in [7], we decided to use more sophisticated dual camera mount by Apec called "Parallax Setting Device" (PSD), as shown in Fig. 2.

Figure 2. Parallax setting device

Stereoscopic effect created by two cameras can be optimized by proper setting of stereoscopic base and convergence angle set between two cameras. PSD is capable of sliding and rotating cameras for setting stereoscopic base and convergence angle respectively. Device allows perfect alignment of optical axes of the two cameras along both vertical and horizontal directions.

## 5. Synchronization

Two cameras mean two independent video streams are created and must be transferred over the network. To remove any unwanted effects on human observer, both streams need to be synchronized when displayed. Synchronizing reflector solves this problem explicitly using timestamps in RTP and RTCP packets. RTP packets include relative time-stamping information which may differ both in time base and time increment for streams with different sources coming even from several applications running on one client computer. Conversion between relative time and absolute time can be performed using information sent in RTCP packet that are sent with much lower frequency for each stream. RTCP packets contain both relative time-stamp and absolute time-stamp in NTP format. Therefore after receiving two RTCP packets it is possible to calculate both relative time base and increment. To synchronize streams coming from two different machines, they must have their clocks synchronized, e. g. using NTP protocol. Conversion between "real" absolute time and relative RTP time based on RTCP information is depicted in Fig. 3.

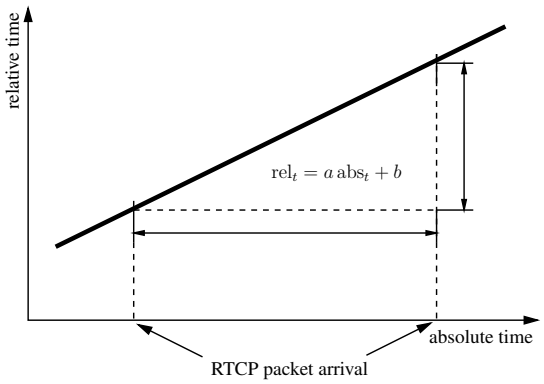



Figure 3. RTP time conversion

For the UDP stream, with no guarantee on delivery, two steps are necessary for the actual synchronization: (1) reorder and/or discard out-of-order packets and (2) match the packets using the RTP/RTCP time information from different streams.

The synchronization and packet reordering is not without penalty. The overall latency increases, which is not desirable for interactive applications like videoconferencing. In such cases even small latency in order of hundreds of milliseconds induces communication problems and disrupts the reality illusion (e. g., when one person tries to interrupt the other one to express his/her

F

opinion).

In order to be able to synchronize RTP streams each RTP packet must have its own timestamp. However, for the DV transmission the timestamps are increased once per a videoframe, which is not acceptable for fine-grained synchronization. To provide synchronization support on a per packet basis, we had to change the protocol and to put individual timestamp into each RTP packet with the DV payload.

## 6. Implementation

While it is possible to synchronize two independent streams only at the point of delivery (before they are displayed), we opted for a network support where the streams are synchronized every time they are retransmitted through a reflector. This solution not only reduces the overall latency due to synchronization and possible reordering, but also provides support to synchronize several sites in one step.

We have enhanced our reflector implementation [8] to support synchronization of multiple reflected RTP packet streams for synchronized, timely, and optionally also in-order delivery to the connected clients. The reflector uses multi-threaded model in which several threads are used as network listeners that place packets coming from different streams into ordered buffers for each stream. Then the sending thread takes packets out of these queues and sends them to the connected clients in synchronized way.

Data processing within the reflector called `rum` proceeds as follows. The reflector starts $N$ separate threads, where $N$ is the number of ports placed as arguments. The main thread is now used as sender and the $N$ threads are used as receiving listeners. Each receiving thread initializes particular socket the reflector is listening to (there are actually two sockets initialized for each RTP session—one for RTP and other one for RTCP packets).

When RTP packet arrives to the listener thread, the RTP header is extracted and parsed to obtain packet relative creation time, which is in turn converted into absolute time. Then the packet is stored into time-sorted buffer—oldest packets are on the top and wait to be sent. Information on stored and dropped packets is kept for statistical purposes.

After receiving an RTCP packet the data for conversion between relative RTP time and absolute time is updated, taking into account previous conversion data by computing sliding average. We assume linear dependence of relative and absolute time. If abrupt change occurs program waits for at least three consecutive RTCP packets carrying consistent time information to achieve stability and avoid short time fluctuations.

The main function of sending thread is to send packets which are saved on top of all the buffers (doing that synchronously if requested). It is also possible to specify that all the late arriving packets are dropped from the buffers. Sending is performed using round robin method for all buffers. Packets on the buffer top are sent to the connected clients when their absolute time is smaller or equals to absolute time of packets in other buffers. When all buffers are empty, the sending process stops and reflector waits for incoming packets. If requested by the user, it is possible to make the reflector stop for random time period after the completion of each cycle (until all buffers are full enough), reducing thus processor load (and naturally increasing overall latency). This option is needed when the reflector is synchronizing low bandwidth data because all buffers have to contain some packets to perform correct synchronization.

## 7. Testbed and Measurements

The reflector implementation, as an implementation of the network supported synchronization of the stereoscopic video streams, has been subjected to a series of tests to evaluate its performance. The goal of these tests was to confirm the synchronization capabilities of the reflector and also to evaluate the additional latency induced by the synchronization effort. DV format was used for both video streams.

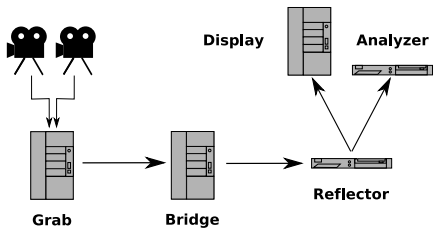



Figure 4. Reflector testbed

The evaluation testbed depicted in Fig. 4 was composed of the following components:

**Sender** FreeBSD 5.4-RC4, CPU: VIA C3 (1200 MHz), 512 MB RAM

**Bridge** Dell 1600SC, FreeBSD 5.3-RELEASE-p8 with two network interfaces, CPU: Intel Xeon (2800 MHz), 1 GB RAM

**Reflector** Linux 2.6.8-2-686-smp, CPU: 2× Intel Xeon (3000 MHz), 4 GB RAM

**Analyzer** Linux 2.6.8-2-686-smp, CPU: 2× Intel Xeon (3000 MHz), 4 GB RAM

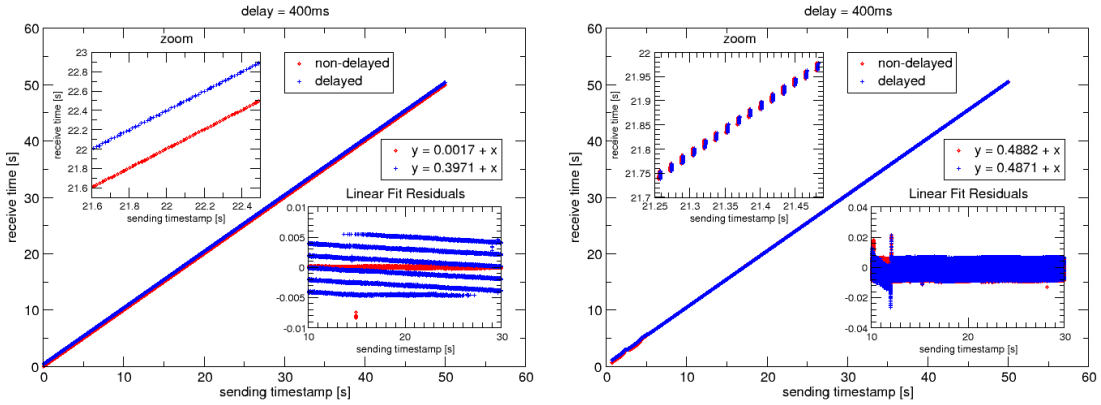



Figure 5. Graph with delay 400 s without/with synchronization

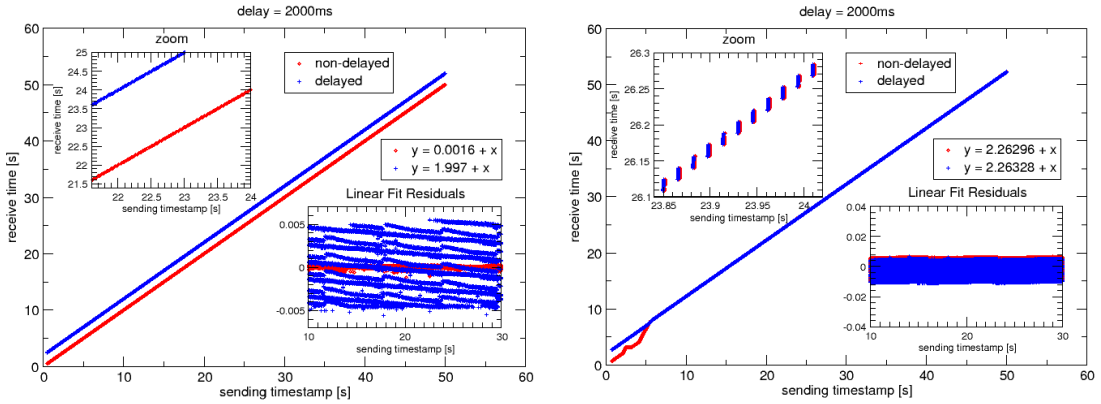



Figure 6. Graph with delay 2000 s without/with synchronization

**Display** Dell Precision 530, Linux 2.4.27-smp, CPU: 2× Intel Xeon (2400 MHz), 1 GB RAM

**Switch** HP Procurve 6108

All the computers used Intel PRO/1000 MT network interface cards, the network link capacity (available bandwidth) has been always 1 Gbps. The testbed used dedicated machines and links.

All measurements were done on two streams where one stream has been sent without any delay and the other has been delayed for a specified time using traffic shaper available as a part of `ipfw` firewall in the FreeBSD kernel. Both streams arrived to the reflector, has been synchronized there and sent to the analyzer (and simultaneously also to the displaying device so we were able to watch the effect also subjectively, but this evaluation is not presented). To cover wide range of possible delays in real production networks, we choose the following delays: 0, 20, 40, 80, 100, 200, 400, 600, 800, 1000, 1500, 2000 and 5000 ms. The choice of measured delays corresponds to the timing of individual fields and frames in 25 fps PAL video. The measured total delay after synchronization is presented in Tab. 1.

We see that the reflector internal delay is around 68 ms, which is only slightly increased if the inter-stream delay is just 20 ms. However, with increased inter-stream delay the total penalty sharply increases. The worst cases are around 100 – 200 ms. With further increase of inter-stream delay the absolute and especially the relative penalty decreases, as the reflector has time to process both streams practically independently. The final penalty for very high inter-stream delay is around 10% of this delay, which is probably acceptable overhead for fully software-based solution.

All the output streams has been fully synchronized. To demonstrate it, we selected two measurements, with

| Link delay [ms] | Synchron. delay [ms] | Link delay [ms] | Synchron. delay [ms] |
|---|---|---|---|
| 0 | 68.3 | 600 | 642 |
| 20 | 71.1 | 800 | 901 |
| 40 | 142 | 1000 | 1130 |
| 80 | 287 | 1500 | 1690 |
| 100 | 357 | 2000 | 2240 |
| 200 | 368 | 5000 | 5550 |
| 400 | 487 | | |

Table 1. Link and synchronization delays

inter-stream delay of 400 and 2000 ms, depicted in Figs. 5 and 6 respectively. All graphs show the relationship between the receiving and sending time of RTP packets. The graphs on the left side show both streams without synchronization—we can see two separate streams (the smaller picture on the left side provides a zoomed view of the relationship). The graphs on the right side show the resulting synchronized stream. The linear-fit residual graphs show the difference between the actual time when a particular RTP packet has been received by the target (the analyzer) and the ideal time of its reception. Again, the results confirm that the synchronization is practically absolute. The initial time synchronization, which needs several RTCP packets, is visible as the small nonlinearity at the beginning of the measurements.

## 8. Conclusions

When stereoscopic video is sent over IP network in two independent streams, they must be synchronized before the they are is displayed. If multiple sites are receiving the same stereoscopic video, the synchronization is best done in the network, otherwise each site may be exposed to different latency, unacceptable for interactive applications.

The idea of overlay network with active elements capable of providing new functionality to computer networks has already been shown as a successful foundation of controlled multicast transmission. The same idea has been used for the stereoscopic video streams synchronization. A simple software implementation running on commodity hardware is able synchronize the two streams in DV format successfully even when the original streams are highly de-synchronized. The penalty of the synchronization is increased latency, as the "faster" stream must wait for data in the other stream, plus some processing latency is added to the final perceived delay. While this delay may be problematic in interactive implementation, the reflector based synchronization element can be easily used for synchronized unidirectional stereoscopic streaming to multiple end users even in highly adverse and desynchronizing network conditions.

## 9. Acknowledgments

This research is supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201) and a research intent "Highly Parallel and Distributed Systems" (MŠM 00216224419). We would also like to thank to Petr Holub for helping with methodology and interpretation of measurements and Luděk Matyska for proof reading this text.

F

## References

[1] Akimichi Ogawa and Katsushi Kobayashi and Kazunori Sugiura and Osamu Nakamura and Jun Muri. *Design and Implementation of DV based video over RTP*. November 2004. `http://www.sfc.wide.ad.jp/DVTS/pv2000/index.html`

[2] H. Schulzrinne and S. Casner and R. Frederick and V. Jacobson. *RFC3550–RTP: A Transport Protocol for Real-Time Applications*. July 2003. `http:///www.zvon.org/tmRFC/RFC3550/Output/index.html`

[3] K. Kobayashi and A. Ogawa and S. Casner and C. Bormann. *RFC3189–RTP Payload Format for DV (IEC 61834) Video*. January 2002. `http://www.zvon.org/tmRFC/RFC3189/Output/index.html`

[4] K. Kobayashi and A. Ogawa and S. Casner and C. Bormann. *RFC3190–RTP Payload Format for 12-bit DAT Audio and 20- and 24-bit Linear Sampled Audio*. January 2002. `http://www.zvon.org/tmRFC/RFC3190/Output/index.html`

[5] V. Griberg and G. Podnar and M. Siegel. *Geometry of binocular imaging*. in Stereoscopic Displays and Applications. February 1994. `http://www.ri.cmu.edu/pub_files/pub1/grinberg_v_s_1994_1/grinberg_v_s_1994_1.pdf`

[6] V. Griberg and G. Podnar and M. Siegel. *Geometry of binocular imaging II: The augmentedeye*. in Stereoscopic Displays and Applications. February 1995. `http://www.ri.cmu.edu/pub_files/pub1/grinberg_v_s_1995_1/grinberg_v_s_1995_1.pdf`

[7] Rhys Hawkins. *Digital Stereo Video: display compression and transmission*. February 2002. `http://escience.anu.edu.au/research/papers/02_Rhys_Hawkins/thesis-small.pdf`

[8] E. Hladká, P. Holub and J.Denemark. *User Empowered Virtual Multicast for Multimedia Communication*. in ICN'2004 Conference Proceedings. March 2004.

# Appendix G

# Active Elements for High-Definition Data Distribution

by Petr Holub, Eva Hladká, Jiří Denemark, and Tomáš Rebok

In 13th International Conference on Telecommunications, ICT'2006, Funchal, Madeira, May 2006. Proceedings. University of Aveiro, Portugal, 2006. 4 p. ISBN 972-98368-4-1.

# Active Elements for High-Definition Video Distribution

Petr Holub*, Eva Hladká†, Jiří Denemark†, and Tomáš Rebok†
*Institute of Computer Science and †Faculty of Informatics
Masaryk University in Brno
Botanická 68a, 602 00 Brno, Czech Rep.
Email: hopet@ics.muni.cz, eva@fi.muni.cz, jirka@ics.muni.cz, xrebok@fi.muni.cz

***Abstract*— Active Elements (AEs), an extension of user-empowered programmable (active) routers, provide support for multimedia distribution in collaborative environments. They can be organized in distributed systems to mitigate the scalability problem for very bandwidth and/or processing power demanding distribution patterns. We present a prototype implementation of Active Elements based on JXTA peer-to-peer framework. As shown in this paper, the AEs are flexible enough to be used for distributing high-definition uncompressed video, while also providing additional processing possibilities when distributing lower bandwidth streams like compressed HDV video streams.**

## I. Introduction

The general problem of synchronous (on-line or interactive) processing lies in providing environment with as low latency of processing and data distribution as possible. We are building user-empowered network support for synchronous multi-point data distribution, that is both scalable to support a large number of clients and also robust with respect to outage of network links and other elements inside the network. This research extends our development of active router and reflectors, providing a general Active Element concept based on similar ideas and principles.

Distribution of multimedia data over IP network leads to a multicast schema. However, as the native multicast solution is not always reliable or even available, other distribution schemes were developed following approach of multicast virtualization. They are usually based on a central distribution unit—a reflector, like the H.323 Multi-point Control Unit (MCU)—that may be organized into a cascade (network) when a higher number of clients needs to be supported. The cascade of H.323 MCUs is usually statically configured and does not offer a user-empowered approach. Another well known example of multicast-like schema is the distribution used in the Virtual Room Videoconferencing System (VRVS) [1]. This is provided as a service and user data traffic is managed by VRVS administrators. The successor of VRVS called Enabling Virtual Organizations (EVO) [2]—is based on self-organization of system of reflectors, again not empowering the end-user with tools to change the distribution topology. There are also other simpler UDP packet reflectors available like rcbridge [3], [4], reflector [5], and Alkit Reflex [6].

The paper is organized as follows: Section II briefs Active Elements (AEs) approach, Section III summarizes our collaborative tool based on high-definition (HD) video, Section IV describes results of using AEs for uncompressed HD video distribution while Section V deals with distribution and processing of compressed HDV streams, and Section VI contains concluding remarks and future work outline.

## II. Active Elements

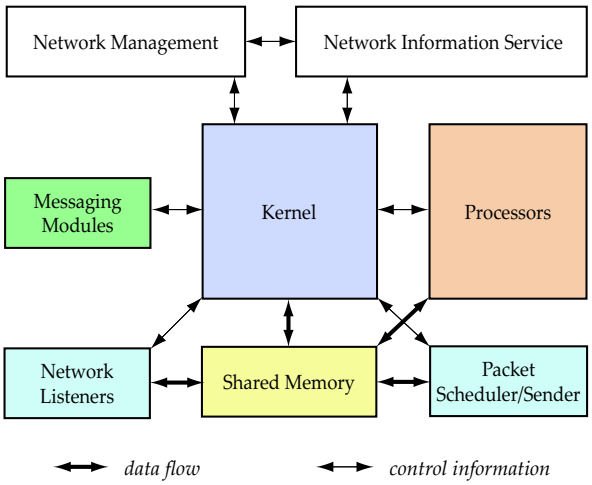
Real-time virtual collaboration needs a synchronous multimedia distribution network that operates at high capacity and low latency. Such a network can be composed of interconnected service elements—so called *Active Elements* (AEs) [7]. They are a generalization of the user-empowered programmable reflector that is a programmable network element replicating and optionally processing incoming data usually in the form of UDP datagrams, using unicast communication only. If the data is sent to all the listening clients, the number of data copies is equal to the number of the clients, and the limiting outbound traffic grows with $n(n-1)$, where $n$ is the number of sending clients. The reflector runs entirely in user-space and thus it works without need for administrative privileges on the host computer, which can be understood as implementation of user-empowered principle.

The AEs add networking capability, i. e. inter-element communication, and also capability to distribute its modules over a tightly coupled cluster. Only the networking capability is important for scalable environments discussed in this paper. The network management is implemented via two modules dynamically linked to the AE in run-time: Network Management (NM) and Network Information Service (NIS) as shown in Figure 1. The NM takes care of building and managing the network of AEs, joining new content groups and leaving old ones, and reorganizing the network in case of link failure. NIS gathers and publishes information about the specific AE (e. g. available network and processing capacity), about the network of AEs, about properties important for synchronous multimedia distribution (e. g. pairwise one-way delay if available or RTT otherwise, estimated link capacity), and also information on content and available formats distributed by the network.

For the out-of-band control messages, the AE network uses self-organizing principles already successfully implemented in common peer to peer (P2P) network frameworks, namely for AE discovery, available services and content discovery, topology maintenance, and also for control channel management. The P2P approach satisfies requirements on both robustness

G

Fig. 1. Architecture of Active Element with Network Management and Network Information Service modules.




and user-empowered approach and its lower efficiency has no significant negative impact as it routes administrative data only. In prototype implementation, this has been implemented using JXTA P2P substrate [8].

### A. Re-balancing and Fail-Over Operations

The topology and use pattern of any network changes rather frequently, and these changes must be reflected in the overlay network, too. We consider two basic scenarios: (1) re-balancing is scheduled due to either use pattern change or introduction of new links and/or nodes, i. e. there is no link or AE failure, and (2) a reaction to a sudden failure. In the first scenario, the infrastructure re-balances to a new topology and then switches to sending data over it. On the contrary, a sudden failure in the second scenario is likely to result in packet loss (for unreliable transmission like UDP) or delay (for reliable protocols like TCP), unless the network distribution model has some permanent redundancy built in. The probability of failure of a particular link or AE is rather small, despite high frequency of failures in global view of large networks. Thus the two fold redundancy might be sufficient for majority of applications, and the redundancy may be increased for the most demanding applications.

### B. Data Distribution in Network of AEs

Separation of control plane from data distribution plane allows for modular implementation of distribution models with different properties. We have studied a number of different data distribution models for the AE network [7], which feature different performance to robustness properties:

- *2D full mesh*—the simplest model which features very high robustness so that AE outage only influences the clients that are directly connected; it also minimizes number of hops inside the overlay network,
- *3D layered mesh*—this model improves performance over the 2D model while retaining the recovery behavior and minimization of number of hops inside the overlay network,
- *3D layered mesh with intermediate AEs*—additional improvement over the 3D layered model, which can be seen as a transition to spanning trees,
- *redundant (minimum) spanning trees*—the model which allows maximum flexibility, efficient recovery from the network outages, allows optimizing data distribution with respect to saturation of lines; extension to multiple precomputed redundant spanning trees brings about capability of very fast recovery.

## III. High-Definition Interactive Collaborative Environments

### A. Uncompressed High-Definition Video Transport

Enabled by current high-speed networks, high-definition (HD) video transmissions have become an essential tool for many applications. Providing truly interactive collaborative environment is still very challenging, because it requires severe limitations on processing to achieve acceptable level of interactivity (ideally less than 100 ms) and thus use of uncompressed video is the most convenient. That however imposes high demands on underlying networking infrastructure especially for multi-point data distribution, as each video stream has 1.5 Gbps. During 2005, we have developed and successfully demonstrated a prototype of low latency multi-site collaborative environment based on uncompressed HD-SDI video according to the SMPTE 292M standard [9].

The whole system comprises two basic parts: client applications and network distribution and processing. We have developed the client tools based on DVS Centaurus[1], Chelsio 10 GE cards and UltraGrid software by Colin Perkins and Ladan Gharai [10], extending it with full 1080i HD support (1920×1080 resolution; previously only lower 720p resolution was supported) and full software display [11] including field de-interlace algorithm and color space down-sampling.

The network distribution requires some service for multi-point distribution in order to supply data from each participant to all the other participants. To remove dependency on native multicast, we are relying on virtual multicast implemented by the AEs. We have demonstrated it's usability over 10 GE networks where each AE was able to duplicate each 1.5 Gbps stream on common dual-AMD64 PC from one partner to two other partners.

### B. HDV Compressed High-Definition Video Transport

HDV [12] is a proprietary MPEG-2 based compression scheme developed by SONY and JVC that is transmitted in the MPEG-TS envelope over the IEEE-1394 interface in a similar way to the DV format for standard definition resolution. The resolution of the video is 1440×1080, with 50 or 60

[1] http://www.dvs.de/english/products/oem/centaurus.html

interlaced fields per second, 8-bit color space, 4:2:0 color space sampling, and 60:1 inter-frame compression resulting in approximately 25 Mbps video stream. We have implemented a tool [13] for FreeBSD 5/6 to read HDV data sent over the IEEE-1394 interface. The data are then rendered using the VideoLAN Client (VLC) tool [14] and sent over the network either using VLC or some other tools like `netcat` [15]. Because of the MPEG-2 format the HDV transmission suffers from higher latency; end-to-end latency measured using laboratory setup described above is as high as 1.9 s, leaving this transport mostly for unidirectional applications like HD streaming only.

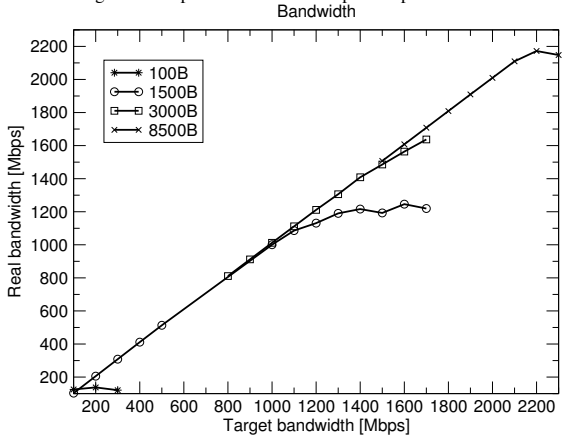
## IV. AEs for Uncompressed HD Distribution

### A. Performance of AEs

We have made several tests to measure the performance of the AE with respect to size of UDP packets used for video transmission. As a testbed we have used two back-to-back connected dual AMD64 computers with 10 GE Chelsio T110 LR NIC cards in 133 MHz PCI-X slots. The following three software components were running on two computers:

- computer A: UDP packet generator providing a UDP stream to computer B at a given bit rate consisting of packets of a desired size,
- computer B: AE receiving the UDP stream from the computer A and replicating it into the two equivalent output streams which are sent back to the computer A,
- computer A: two packet analyzers running in parallel capable of detecting lost and out-of-order packets and average bit rate received from AE on the computer B; each analyzer examines one stream.

Fig. 2. AE performance with respect to packet size.




Performance measurement results of the AE are shown in the Figure 2. Exact numbers describing some key points from the graph are summarized in Table I: (a) gives maximum bandwidth of a single stream with packet loss <0.01% with respect to packet size in use, while (b) gives packet loss with respect to bandwidth of a single stream given 8,500 B packet size, which we have chosen as a standard for our uncompressed HD application. The performance evaluation confirms the necessity to use Jumbo Ethernet frames for the AE to be able to replicate uncompressed HD-SDI video streams at 1.5 Gbps. Setting appropriate MTU on all the hosts and all over the path is the only part that requires administrative privileges, thus violating the user-empowered paradigm.

TABLE I
Performance of the Active Element.

(a)

| *packet size* [B] | *max. bandwidth* [Mbps] |
|---|---|
| 100 | 100 |
| 500 | 300 |
| 1500 | 400 |
| 3000 | 800 |
| 6000 | 1700 |
| 9000 | 2000 |

(b)

| *bandwidth* [Mbps] | *packet loss* [%] | *CPU load* [%] |
|---|---|---|
| 1800 | 0.0 | 52 |
| 1900 | 0.0 | 55 |
| 2000 | 0.0099 | 60 |
| 2100 | 0.037 | 76 |
| 2200 | 1.74 | 80 |
| 2300 | 7.07 | 84 |

### B. Real-World Performance

The HD transport together with AE for data distribution has been demonstrated during iGrid 2005 workshop [11] (CZ101 and US127 demos) and at SC 2005 conference. Because no more than 3 sites were participating, there was no real need to build the network of AEs. However, in order to demonstrate feasibility of this approach, we have also built a 3D layered mesh of AEs with one intermediate AE where AE cascading was used for multiplicating one stream to one site in multiple copies, so that one site saw one stream multiple times. Because a single AE running on a high-end machine is unable to create more than two copies of the data at 1.5 Gbps because of traffic limitations on given hardware, a AE network is required for supporting more than 3 sites, which is not very efficient.

We also had to solve a problem with bursty HD traffic, which lead to loss of many packets due to the implicit thread switching. The original AE implementation used two different threads—one for the network listener and the second for sender. With implicit thread switching, the listener thread could have been stopped and the sender thread activated regardless of whether more data were coming. The current implementation gives explicit precedence to the listener which yields its precedence to the sender only when no more data are to be read. The only negative outcome we have observed with this approach is higher CPU load on the AE.

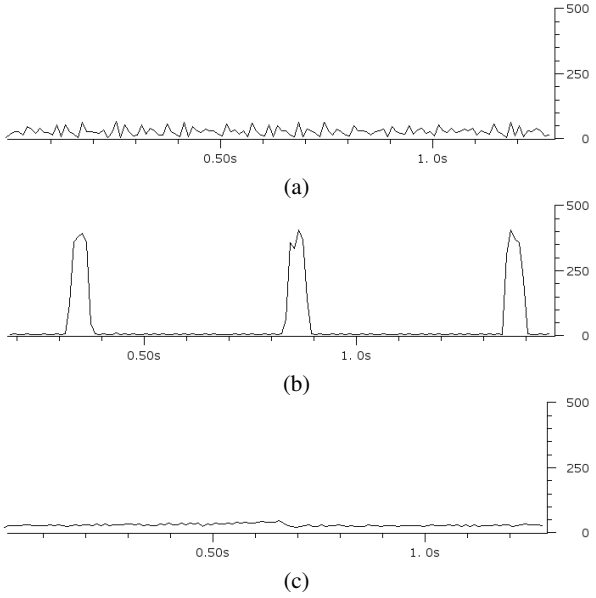
## V. AEs for HDV Distribution

The high processing capacity of the AE can be also used for other purposes, e. g. for traffic shaping in case of very bursty

data distribution with otherwise only moderate bandwidth requirements. We demonstrate this potential on the distribution of bursty HD traffic in the HDV format.

The HDV stream is highly compressed, requiring only 25 Mbps bandwidth. However, the HDV data generated by commonly used tools are very bursty, with peeks almost 20 times above the theoretical bandwidth requirements. Burstiness is introduced by the sending application—either VLC (Figure 3a) or even worse using `netcat` (Figure 3b).

Fig. 3. Bursty traffic produced by HDV transport with VLC (a) and `netcat` (b). Traffic after smoothing by AE is shown in (c). Horizontal axes show time and vertical axes show number of packets received at each time instant.




We have enabled a traffic smoothing module in the AE for smoothing based on sliding average of the incoming bandwidth. The resulting data flow is very smooth with no bursts at all as shown in Figure 3c. The penalty for this is small increase of transmission latency (depending on burst characteristics and smoothing interval, ranging from tens to hundreds milliseconds) because of packet buffering. The observable advantage of having smooth data flows is the receiving VLC being capable of rendering the stream with close to zero image defects, which is impossible for bursty streams above 10 Mbps.

## VI. Conclusion

In this paper the Active Element and networks of AEs have been introduced to provide a flexible multi-point data distribution environment. We have discussed the advantages of this solution and we have described the processing power, robustness and fail-over capabilities.

Further, the AEs were used to support a simple collaborative environment based on the HD video transmission. The flexibility of our approach has been demonstrated on using the same elements (only with different configuration) for the distribution of uncompressed HD video streams with 1.5 Gbps stressing the data duplication capacity of the AEs, and for the distribution of bursty HDV traffic, where the smoothing extreme traffic bursts again shown processing capacity of the AEs—in this case as traffic shapers. Though not very efficient, we also demonstrated that AE network can support higher number of clients even with the uncompressed HD streams.

The future work will be focused on supporting real multi-point distribution of high speed multimedia streams. This is required when a larger number of clients must be connected, either because of higher number of participating sites or because each site sends multiple HD video streams (e. g. visualization and videoconferencing, stereoscopic streams etc.). We will also continue developing the concept of distributed AE to provide scalable multi-point replication of high demanding data.

## Acknowledgment

This research is supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201). We would also like to thank to Michal Procházka, Miloš Liška and Lukáš Hejtmánek for their help with implementation of HD transport tools.

## References

[1] "Virtual Room Videoconferencing System (VRVS)," http://www.vrvs.org/.
[2] P. Galvez, "From VRVS to EVO (Enabling Virtual Organizations)," in *TERENA Networking Conference 2006*, Catania, Italy, May 2006, *Accepted.*
[3] M. Buchhorn, "Designing a multi-channel-video campus delivery and archive service," in *The 7th Annual SURA/ViDe Conference*, Atlanta, GA, USA, Mar. 2005.
[4] "rcbridge," http://if.anu.edu.au/SW/rcbridge.html.
[5] O. Hodson, "UDP packet reflector/forwarder," http://www.cs.ucl.ac.uk/staff/s.bhatti/teaching/z02/reflector.html.
[6] "Alkit reflex," http://w2.alkit.se/reflex/.
[7] P. Holub, E. Hladká, and L. Matyska, "Scalability and robustness of virtual multicast for synchronous multimedia distribution," in *Networking - ICN 2005: 4th International Conference on Networking, Reunion Island, France, April 17-21, 2005, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 3421/2005. La Réunion, France: Springer-Verlag Heidelberg, Apr. 2005, pp. 876–883.
[8] P. Holub and E. Hladká, "Ubiquitous user-empowered networks of active elements," in *TERENA Networking Conference 2005*, Poznań, Poland, June 2005, p. 3.
[9] Society of Motion Picture and Television Engineers, "Bit-serial digital interface for high-definition television systems," SMPTE 292M-1998.
[10] C. Perkins, L. Gharai, T. Lehman, and A. Mankin, "Experiments with delivery of HDTV over IP networks," in *12th International Packet Video Workshop*, Pittsburgh, PA, USA, Apr. 2002.
[11] P. Holub, L. Matyska, M. Liška, L. Hejtmánek, J. Denemark, T. Rebok, A. Hutanu, R. Paruchuri, J. Radil, and E. Hladká, "High-definition multimedia for multiparty low-latency interactive communication," *Future Generation Computer Systems*, 2006, *Accepted*.
[12] "HDV information," http://www.hdv-info.org/.
[13] P. Holub, "HDV capture for FreeBSD 5/6 operating system," http://sitola.fi.muni.cz/~hopet/HDV/ and http://docs.freebsd.org/cgi/getmsg.cgi?fetch=37672+0+archive/2005/freebsd-firewire/20050206.freebsd-firewire.
[14] "VideoLAN Client (VLC)," http://www.videolan.org/.
[15] "netcat," http://netcat.sourceforge.net/.

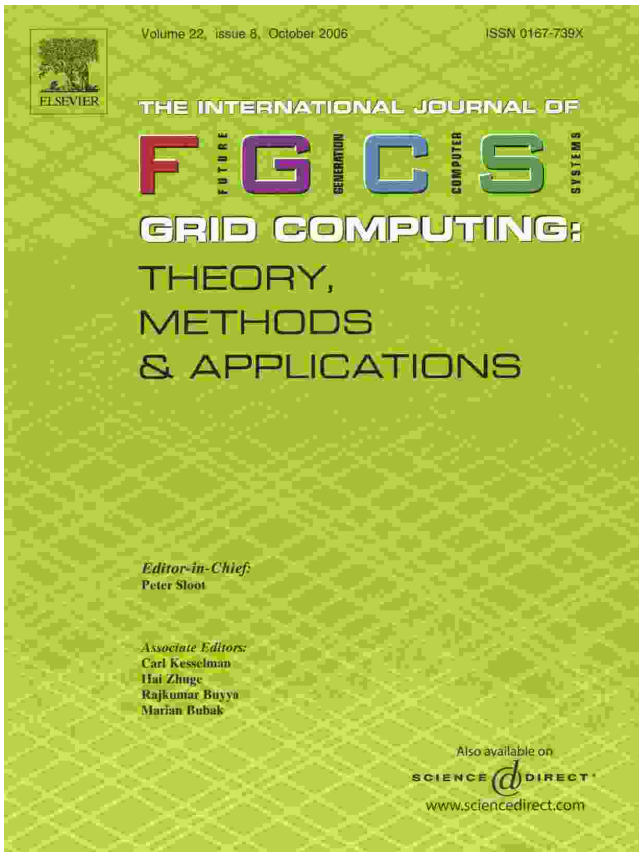
G

# Appendix H

# High definition multimedia for multiparty low-latency interactive communication

by Petr Holub, Luděk Matyska, Miloš Liška, Lukáš Hejtmánek, Jiří Denemark, Tomáš Rebok, Andrei Hutanu, Ravi Paruchuri, Jan Radil and Eva Hladká

Future Generation Computer Systems, Amsterdam, The Netherlands: Elsevier Science, 22, 8, pp. 856–861, 6 p. ISSN 0167-739X. 2006.



Available online at www.sciencedirect.com
SCIENCE DIRECT®
Future Generation Computer Systems 22 (2006) 856–861
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

# High-definition multimedia for multiparty low-latency interactive communication

Petr Holub[a,b], Luděk Matyska[a,b,*], Miloš Liška[a,b], Lukáš Hejtmánek[a,b], Jiří Denemark[a,b], Tomáš Rebok[a,b], Andrei Hutanu[c], Ravi Paruchuri[c], Jan Radil[a], Eva Hladká[a,b]

[a] *CESNET z.s.p.o., Zikova 4, 16200 Praha, Czech Republic*
[b] *Masaryk University, Botanická 68a, 62100 Brno, Czech Republic*
[c] *Center for Computation and Technology, 302 Johnston Hall, Louisiana State University, Baton Rouge, LA 70803, United States*

Available online 30 May 2006

**Abstract**

We describe a high-quality collaborative environment that uses High-Definition (HD) video to achieve near realistic perception of a remote site. The capture part, consisting of a HD camera, Centaurus HD-SDI capture card, and UltraGrid software, produces a 1.5 Gbps UDP data stream of uncompressed HD video that is transferred over a 10GE network interface to the high-speed IP network. The HD video stream displaying uses either a software-based solution with color depth down-sampling and field de-interlacing, or another Centaurus card. Data distribution to individual participants of the videoconference is achieved using a user-controlled UDP packet reflector based on the Active Element idea. The viability of this system has been demonstrated at the iGrid 2005 conference for a three-way high quality videoconference among sites in the Czech Republic, Louisiana, and California.


*Keywords:* High-Definition Video; Video Conference; Uncompressed HD Video; High-Speed Multipoint Data Distribution

## 1. Introduction

Evolution of collaborative environments, following the development of high-band-width low-latency networks brings new possibilities to the quality and extent of collaboration [1,2]. The truly interactive communication requires high-resolution video and audio transmitted fast over a network, with end to end latency below 100 ms to avoid hearing artifacts. Even if captured and transmitted independently, the video and audio must be kept synchronized and thus both transmitted with the lowest latency. All components of an end to end path contribute to the latency, so using uncompressed media is essential, together with almost no buffering. The network needs to provide support by close to zero packet reordering and loss, low latency, and minimal jitter. Such a network can be efficiently constructed based on dedicated circuits over a fiber optic network.

In this paper we describe a multiparty high-definition (HD) videoconferencing system and experiences gained from its use during the iGrid 2005 conference.

## 2. HD video transport and distribution

The highest resolution for the HD video defined by the common HDTV standard [3] is the 1080i mode with 1920 × 1080 points and interlaced line scanning. We use the uncompressed HD digital video defined in SMPTE 292M [4], which is transmitted through the Serial Digital Interface (HD-SDI). The bandwidth for such a video-stream with 60 interlaced fields per second, 10 bits per color plane, and 4:2:2 color space sampling is 1.485 Gbps. This payload is equivalent to around 1.5 Gbps over the IP network with 44 byte header per packet.

* Corresponding address: Institute of Computer Science, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic. Tel.: +420 549492105; fax: +420 541212747.
*E-mail addresses:* hopet@ics.muni.cz (P. Holub), ludek@ics.muni.cz (L. Matyska), xliska@fi.muni.cz (M. Liška), xhejtman@fi.muni.cz (L. Hejtmánek), jirka@ics.muni.cz (J. Denemark), xrebok@fi.muni.cz (T. Rebok), ahutanu@cct.lsu.edu (A. Hutanu), ravi9@cct.lsu.edu (R. Paruchuri), radil@cesnet.cz (J. Radil), eva@fi.muni.cz (E. Hladká).

0167-739X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2006.03.014

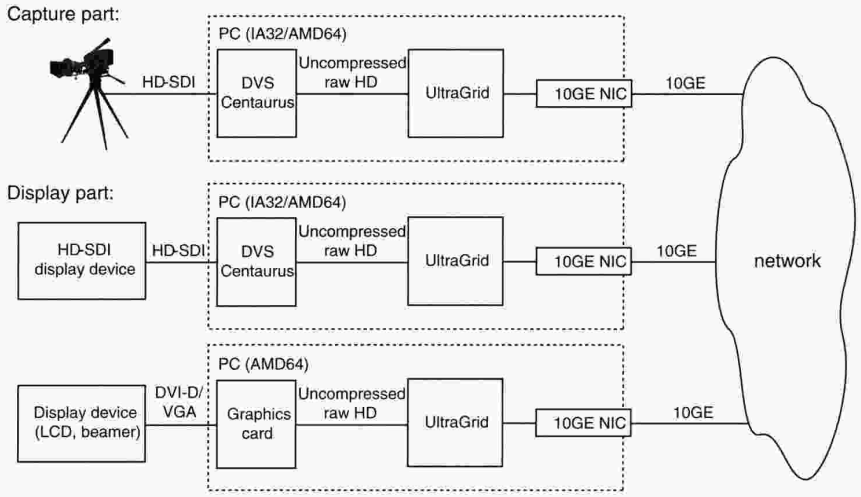



Fig. 1. Site connection scheme for HD transmission.

The capture and display parts of the system able to generate and deal with such data rate are depicted in Fig. 1. The capture part uses the DVS Centaurus (http://www.dvs.de/english/products/oem/centaurus.html) HD capture card, a solution dictated by the selection of Linux operating system environment. We have rewritten the UltraGrid software package [5] to support the 1080i mode. While the same path can be also used for video display (in reverse order), we have extended the UltraGrid with support for software only display, including a field de-interlace algorithm and color space down-sampling from 10 to 8 bits per color plane to avoid the use of expensive Centaurus cards on the receiving end of each HD video path. The computation and data manipulation intensive parts of the UltraGrid software have been optimized for deployment on AMD64 (Opteron) based computers.

The whole capture part starts with the Sony HVR-Z1E camera whose analog output is converted to HD-SDI using the AJA HD10A converter (http://www.aja.com/hd10a.htm). The HD-SDI stream is captured by the DVS Centaurus card, encapsulated into the UDP/IP stream by the Ultra Grid and sent via the Chelsio T110 LR 10GE card (http://www.chelsio.com/T110.htm) to the network. On the receiving end, the IP stream is captured by the same 10GE card, stripped of the IP header, color down-sampled and de-interlaced by the UltraGrid and sent to the graphic card. We use dual AMD64 Opteron 250 computers running at 2.4 GHz with 4 GB RAM and Centaurus and Chelsio cards placed in the PCI-X 133 MHz slots. Linux kernel 2.6.6 is used with drivers and manufacturer provided patches for both cards.

The total end to end latency measured in a laboratory set-up has been 175 ± 5 ms with both computers on the same 10GE Cisco Catalyst 6506 switch. While being above the optimum 100 ms threshold, 175 ms is still acceptable and hardly noticeable on the level of human perception. We have also analyzed part-wise latencies: four fields buffered on the capture card[1] (66.7 ms, calculated), 10 b to 8 b down-sampling (7 ± .5 ms, measured in software), de-interlacing (7 ± .5 ms measured in software), software display (41 ± .5 ms, measured in software), LCD display delay (25 ms, according to manufacturer's specifications). This counts up to 147 ms and we are attributing the remaining delay (28 ms) to in-camera processing, HD-SDI conversion, buffering, and the processing on the graphics card. It is worth noting that the duration of down-sampling and de-interlacing is limited by the memory copying speed (approx. 1 GBps on test-bed machines). The sender and receiver CPU load has been 25% and 72% on average, respectively.

While the audio latency is the main driver for the use of uncompressed video streams, audio does not generate data flows of comparable bandwidth. In our application, the audio stream is generated by the `rat` tool [6] and synchronized on the receiving machine via the capability of UltraGrid to utilize time information from the RTP/RTCP packets.

The HD video may also be transmitted in the compressed HDV format at the cost of latency increase. HDV is a proprietary MPEG-2 based compression scheme developed by Sony and it has been designed to be transmitted in the MPEG-TS envelope over the IEEE-1394 interface, similar to the DV format transmission. It uses only 1440 × 1080 resolution, 50 or 60 interlaced fields per second, 8-bit color space, 4:2:0 color space sampling, and interframe 60:1 compression resulting in approximately 25 Mbps video stream. We have implemented a tool for the FreeBSD operating system [7] to read out the HDV data from the IEEE-1394 encapsulation. The data is sent over the network either using VideoLAN Client (VLC) (http://www.videolan.org/) or some other tool like `netcat` (http://netcat.sourceforge.net/) and it may be rendered using a VLC tool.

The measured end to end latency for the 60i HDV stream in the same laboratory set-up as mentioned above has been 1907 ± 13 ms. As the delay of camera compression is below 1 s, the most of the total latency is accumulated in the VLC buffering, decompression and rendering process. The almost

[1] The 2 fields reported by DVS did not work.

H

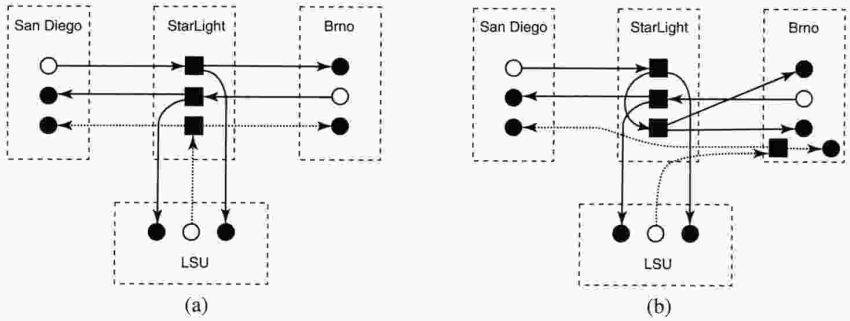



Fig. 2. Connection scheme for (a) the first and (b) the second iGrid experiment. The full lines show uncompressed HD streams; the dotted line shows compressed HDV streams. The empty circles denote sending computers, the full circles stand for display computers, and the full boxes are AEs.

Table 1
Reflector performance on the testbed

| (a) | | (b) | | |
|---|---|---|---|---|
| Packet size (B) | Max. bandwidth (Mbps) | Bandwidth (Mbps) | Packet loss (%) | CPU load (%) |
| 100 | 100 | 1800 | 0.0 | 52 |
| 500 | 300 | 1900 | 0.0 | 55 |
| 1500 | 400 | 2000 | 0.0099 | 60 |
| 3000 | 800 | 2100 | 0.037 | 76 |
| 6000 | 1700 | 2200 | 1.74 | 80 |
| 9000 | 2000 | 2300 | 7.07 | 84 |

(a) Gives maximum bandwidth of a single stream with respect to packet size in use given packet loss <0.01%; (b) gives packet loss with respect to bandwidth of a single stream given 8500 B packet size.

2 s latency practically hinders any real time communication. If it needs to be used, the audio stream must be independent and should not be synchronized with the video. This way, acceptable interactivity level is guaranteed via audio while video visibly lags behind.

To distribute data among more locations in a multipoint videoconference, we used the generalized Active network Element (AE) [8] based on the UDP packet reflector design [9]. This gives us more control over the distribution than a network native scheme (multicast or broadcast), which also may be unavailable for very high-speed networks.

We have optimized the AE to provide a sustainable UDP packet replication rate of up to 2.0 Gbps on high end IA-32 or AMD64 computer. The actual performance on the dual Opteron 250 computer (the same that has been used for the video capture and display) is summarized in Table 1. The results confirm the necessity to use Jumbo frames (long MTU) to achieve the highest performance. Also, the network card to memory bandwidth is important, as the use of an only 100 MHz PCI-X slot reduced the usable reflecting capability to 1.7 Gbps (this is the measured limit above which packets started to be lost significantly). Total latency increase induced by the replication measured on the test bed described above was $13 \pm 2$ ms.

## 3. iGrid 2005 experiment

The goal of the CZ101 iGrid 2005 experiment was providing a low-latency multiparty collaborative environment with HD video. Three sites participated: iGrid premises at San Diego UCSD campus, Masaryk University/CESNET in Brno, Czech Republic, and Louisiana State University (LSU) in Baton Rouge. Three networking circuits (in fact L3 networks) met at StarLight, Chicago, each coming from one participating site: an iGrid circuit from San Diego (RTT $78.2 \pm .2$ ms, 4 hops), an NLR circuit from LSU (RTT $31.09 \pm .04$ ms, 2 hops), and a circuit from Brno spanning CzechLight and NetherLight (RTT $126.7 \pm .3$ ms, 2 hops).

The video capturing and displaying set-up described above was used at all the sites with the exception of LSU, where the unavailability of the Centaurus card[2] led to the fallback HDV solution. The video was displayed using $1920 \times 1200$ resolution at all sites (HD video was wrapped with black borders on top and bottom of the screen, while the HDV had to be up-scaled first).

Three reflectors were set up at StarLight, each replicating a stream from one site – two 1.5 Gbps and one 25 Mbps streams – to the remaining two sites (see Fig. 2(a)). To stress the infrastructure, in the second part of the demo we cascaded two of the reflectors so that Brno was receiving two identical streams from San Diego (see Fig. 2(b)). This way we achieved total network flow of 4.5 Gbps on the Brno–StarLight circuit. This set-up has also proven the feasibility of combining AEs into an AE network to provide scalable data distribution [8] even at very high data rates.

The total unidirectional end-to-end latency for the uncompressed HD distributed using a single reflector was 290 ms in the worst case (San Diego–Brno), while in the best case (San Diego–LSU) it was 242 ms. The total latency was noticeable during the experiment, but it was barely disturbing.

The WAN-PHY interface of the Cisco Catalyst 6506 switch in Brno was used to gather network statistics shown in Fig. 3 during the last demonstration. This was the uplink interface that aggregated all the traffic from and to the Brno site. The outgoing traffic was rather regular, while the incoming traffic displayed some burstiness. We attribute this behavior to the use of the L3 network, where the packets are still buffered at some intermediate network elements and thus the latency is rather uneven. We expect utilization of a pure end to end optical network with no intermediate buffers to remove this problem.

[2] It has been stuck at US customs because of hurricanes.

H

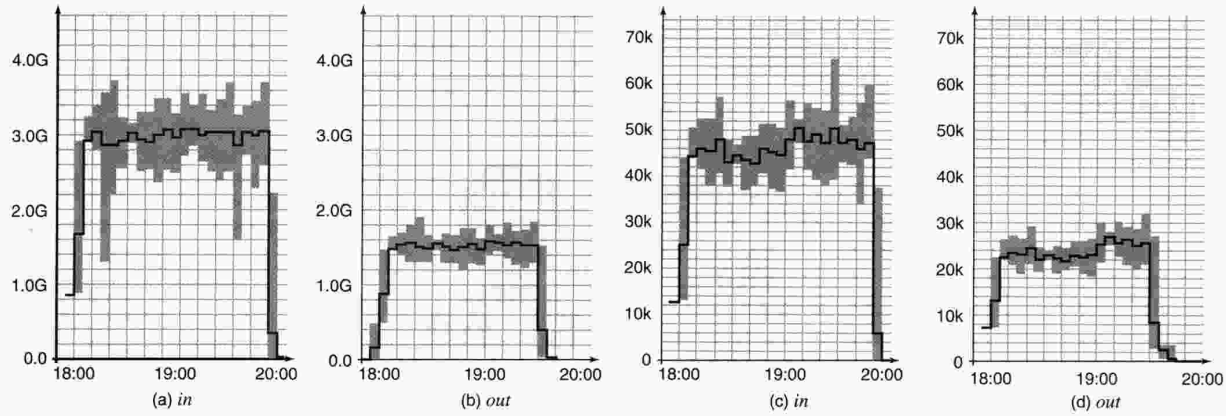



Fig. 3. Network statistics gathered on the Brno uplink. The lines are five minute averages, while the surface is an envelope curve with min/max values within the five minute interval. (a) gives *in* throughput in Gbps, (b) gives *out* throughput in Gbps, (c) gives *in* throughput in packets per second (pps), and (d) gives *out* throughput in pps.

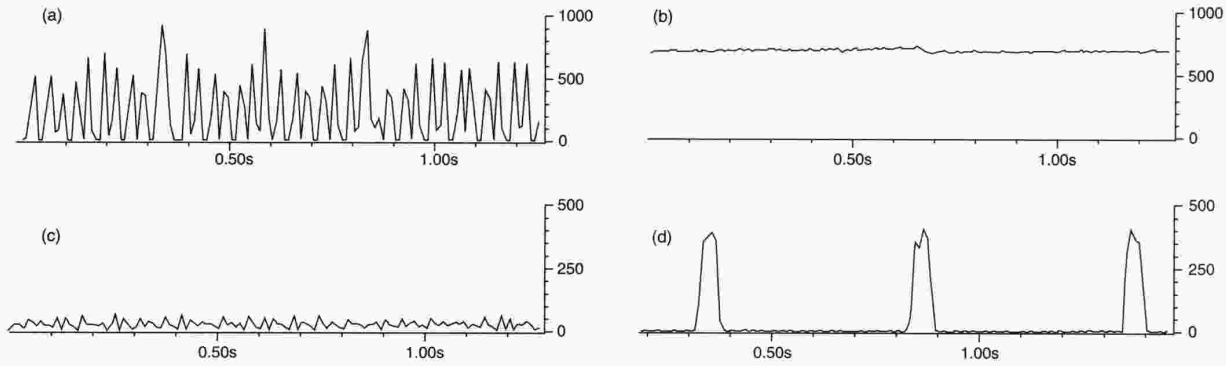



Fig. 4. Bursty traffic produced by the uncompressed HD transport (a) compared to smooth traffic by our measurement probes (b). Bursty traffic produced by the HDV transport with VLC (c) and `netcat` (d). Vertical axis denotes number of packets per 20 ms interval.

## 4. Experiences, problems, and related work

The UltraGrid produces very bursty traffic (see Fig. 4), that reduced the duplicating capacity of the AE to approximately 200–600 Mbps with large fluctuations; no packet loss is observed inside the AE software, as the data was actually lost during reading from the received packet queues of the underlying operating system. Similar burstiness occurs also with the HDV transport as shown in Fig. 4. We modified the AE to use the non-blocking `read()` function giving explicit precedence to the sending thread when no input data is available. While this increased CPU load to 100%, the duplication runs at the target speed.

Another source of problems we encountered was the overheating and instability of dual Opteron computers with Chelsio and especially with both Chelsio *and* Centaurus cards. The fast assemblage and set-up of dual Opteron computers at San Diego even resulted in one of the machines being unable to receive or send data with the Chelsio card above 800 Mbps (the same card worked perfectly if moved into any of the remaining two machines).

**Related Work.** Our work follows development of UltraGrid [5], enhancing it with 1080i resolution support and extending its software display including de-interlacing and color space scaling. We also further extend our work on Active Elements for the efficient data distribution under strict user control (see discussion in [8]). During the iGrid 2005, ResearchChannel had a similar demonstration [10] of multipoint HD videoconference. In contrast to their set-up, we used the software display instead of HD-SDI, independent audio stream (externally synchronized), and Active Elements instead of multicast for data distribution.

## 5. Conclusions and future work

We have demonstrated that a high-quality multiparty videoconference based on transmission of uncompressed HD streams is already achievable provided adequate networking resources are available. The same data distribution and HD video transmission set-up has also been used in combined visualization demo during the conference. Although utilizing software reflector, L3 network instead of dedicated optical circuits, and excessive frame buffering on Centaurus capture card led to higher than theoretically minimal latency and jitter, the whole set-up provided a realistic high-quality collaborative environment.

However, the experience has also demonstrated deficiencies and open challenges suggesting future research and development. At the network level, we plan to repeat the experiment over L2 and L1 networks to achieve smaller latency and jitter [11]. We also plan to replace software based Active Elements with optical splitters that would further reduce the latency for multiparty transmission at the L1 level. The rather complicated set-up of the network for the demonstration also proved the necessity for a special control plane over (optical) networks spanning several administrative domains if dedicated circuits are to be provided effectively. On the application level, we will focus on employing hardware with lower buffering requirements for latency reduction and on implementing smoothing algorithms for packet sending to mitigate problems with bursty traffic.

## Acknowledgments

We would like to thank Tom Košnar, Martin Míchal and Josef Vojtěch from CESNET for network statistic gathering and European optical line provisioning, and Alan Verlo, Pieter de Boer, Paola Grosso and other people at iGrid 2005 NOC for their support. Also, the help of Cisco and Chelsio (both lending some equipment) is highly appreciated. This project has been supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201) and "Parallel and Distributed Systems" (MŠM 0021622419).

## References

[1] T.A. DeFanti, M.D. Brown, C. de Laat, iGrid 2002: The international virtual laboratory, Future Generation Computer Systems 19 (6) (2003) 803–804.
[2] R. Singh, J. Leigh, T.A. DeFanti, F. Karayannis, TeraVision: A high resolution graphics streaming device for amplified collaboration environments, Future Generation Computer Systems 19 (6) (2003) 957–971.
[3] Society of Motion Picture and Television Engineers, 1280 × 720 scanning, analog and digital representation and analog interfaces, SMPTE 296M-1998.
[4] Society of Motion Picture and Television Engineers, Bit-serial digital interface for high-definition television systems, SMPTE 292M-1998.
[5] C. Perkins, L. Gharai, T. Lehman, A. Mankin, Experiments with delivery of HDTV over IP networks, in: 12th International Packet Video Workshop, Pittsburgh, PA, USA, 2002.
[6] V. Hardman, A. Sasse, M. Handley, A. Watson, Reliable audio for use over the internet, in: Proceedings of INET'95, Honolulu, Hawaii, 1995.
[7] P. Holub, HDV capture for FreeBSD 5/6 operating system. http://sitola.fi.muni.cz/~hopet/HDV/, http://docs.freebsd.org/cgi/getmsg.cgi?fetch=37672+0+archive/2005/freeb%sd-firewire/20050206.freebsd-firewire.
[8] P. Holub, E. Hladká, L. Matyska, Scalability and robustness of virtual multicast for synchronous multimedia distribution, in: Networking—ICN 2005: 4th International Conference on Networking, 17–21 April, Reunion Island, France, 2005, Proceedings, Part II, in: Lecture Notes in Computer Science, vol. 3421/2005, Springer-Verlag, Heidelberg, La Réunion, France, 2005, pp. 876–883.
[9] E. Hladká, P. Holub, J. Denemark, An active network architecture: Distributed computer or transport medium, in: Proceedigns of 3rd International Conference on Networking, ICN'04, Gosier, Guadeloupe, 2004, pp. 338–343.
[10] M. Wellings, J. DeRoest, A. Philipson, J. Eveleth, J. Brown, C. Latham, R. Johnson, G. McLaughlin, A. Howard, J. O'Callaghan, M. Lack, E. Verharen, D. Devereaux-Weber, A. Kato, Global N-way interactive high-definition video conferencing over long-pathway, high-bandwidth networks, US118 iGrid 2005 demo. http://www.igrid2005.org/program/applications/videoservices_nwayconf.html.
[11] C. de Laat, E. Radius, S. Wallace, The rationale of the current optical networking initiatives, Future Generation Computer Systems 19 (6) (2003) 999–1008.

**Petr Holub** graduated at Faculty of Sciences, Masaryk University in Brno and received the Ph.D. in computer science from Faculty of Informatics MU in informatics, focusing at high-speed networks, multimedia, and parallel and distributed systems. Currently he works at Institute of Computer Science MU in the Laboratory of Advanced Networking Technologies and participates on its scientific leadership. He is also a researcher with CESNET. His professional interests include high-speed networks and suitable protocols, active networks, user-empowered overlay networks, advanced collaborative environments, grid environments, and computational quantum chemistry and its implementation on distributed systems.

**Luděk Matyska** is an Associate Professor in Informatics at Faculty of Informatics, and he also serves as a vice-director of Institute of Computer Science, both at Masaryk University in Brno, Czech Republic. He got a Ph.D. in Chemical Physics from Technical University Bratislava, Slovakia. His research interests lie in the area of large distributed computing and storage systems, with a specific emphasis on their management and monitoring. He also works in high speed network applications, with a specific emphasis on collaborative work support and use of all these technologies in various e-learning activities. He lead national Grid infrastructure projects and participates in several EU funded international projects including the CoreGRID Network of Excellence.

**Miloš Liška** is a Ph.D. student at Faculty of Informatics at Masaryk University. He is also a CESNET researcher working on projects concerning multimedia, tools for collaborative environments and collaborative workflows. He is interested especially in tools for multimedia processing and video transmission.

**Lukáš Hejtmánek** graduated in computer science and currently he is a Ph.D. student at Faculty of Informatics, Masaryk University. His main research interests include: high speeds networks, network data storage and peer to peer storage, distributed storage with client–server semantics, data replicas, replicas management, and data indices.

**Jiří Denemark** graduated in computer science at Faculty of Informatics, Masaryk University. Currently he is a Ph.D. student at Masaryk University and participates in CESNET research projects focused on multimedia data transmissions and processing and environment for virtual collaboration. His research interests are active overlay networks, virtual multicast, network support for collaborative systems, virtual machines and utilizing their technologies in grid environments.

**Tomáš Rebok** graduated in computer science at the Faculty of Informatics at the Masaryk University in Brno and currently he is a Ph.D. student at the same faculty. He is also a researcher with CESNET, working on problems of synchronous multimedia distribution and network performance measurement technologies. His research interests include active routers and distributed routers with QoS support on high-speed networks.

H

**Andrei Hutanu** graduated in 2002 and received his engineering diploma in Computer Science from Politehnica University of Bucharest. Currently he is a researcher in the Grid and Visualization departments of the Center for Computation and Technology at Louisiana State University. Prior to this he was employed for two years as a researcher in the Visualization department of the Zuse Institute Berlin. His research interests are distributed interactive visualization, data management and optical network applications.

**Ravi Paruchuri** received his B.Tech. from University of Madras in computer science and engineering and his M.Sc. from Louisiana State University in systems science. Currently he is the manager of IT Operations within the Center for Computation and Technology at Louisiana State University.

**Jan Radil** received the M.Sc. and Ph.D. degrees in electrical engineering and electrotechnology from the Czech Technical University, Prague, in 1996 and 2004, respectively. He joined the Research and Development Department, CESNET, Prague, in 1999, where he is responsible for optical networking and development of next generation of the experimental research networks.

**Eva Hladká** is an Assistant Professor in Informatics at Faculty of Informatics, she is a head of Laboratory of Advanced Networks Technology. She is also a senior researcher with CESNET. She got Ph.D. in Computer Science from Masaryk University in Brno. Her research interests include user empowered network technology and active networks, network support for collaborative environments, virtual multicast, and P2P networks of communication elements. She also works on collaborative environments for e-learning and telemedecine.

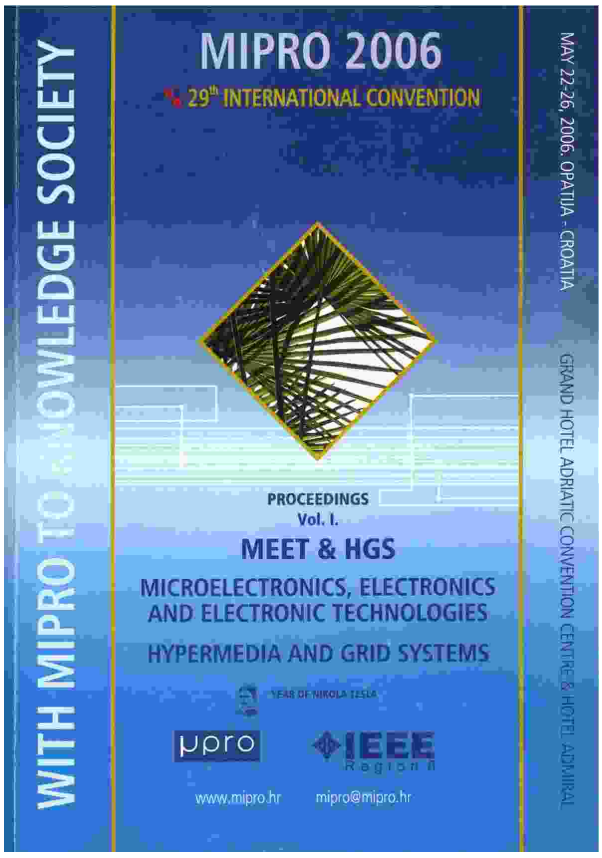
H

# Appendix I

# Quality of Service oriented Active Router Design

by Tomáš Rebok, Petr Holub and Eva Hladká

Microelectronics, Electronics and Electronic technologies, Hypermedia and GRID Systems, MIPRO 2006, Opatija, Croatia, May 2006. Proceedings. Croatian Society for Information and Communication Technology, Electronics and Microelectronics, 2006. 6 p. ISBN 953-233-018-6.



# Quality of Service Oriented Active Routers Design

Tomáš Rebok*, Petr Holub‡, and Eva Hladká*
*Faculty of Informatics and ‡Institute of Computer Science
Masaryk University
Botanická 68a, 602 00 Brno
E-mail: xrebok@fi.muni.cz, hopet@ics.muni.cz, eva@fi.muni.cz

***Abstract*— The active network approach allows an individual user to inject customized programs into the active nodes in the network, usually called programmable/active routers, and thus process data in the network as it passes through. When a programmable router is used in a multi-user network environment, quality of service (QoS) for each passing stream needs to be ensured. QoS approaches in common networks enforce certain parameters (e.g., queuing strategy, priority) on the network flows. However, this is not sufficient in active routers where users' programs run on the routers and thus other parameters (e.g. processor time, amount of memory) have to be guaranteed as well. In this paper, we propose a QoS-enabled active router architecture that supports extended understanding of QoS. We also propose a virtual machine based router implementation for strict isolation of user processes.**

## I. Introduction

Contemporary computer networks behave as a passive transport medium which delivers—or in case of best-effort service tries to deliver—data sent from the sender to the receiver. The whole transmission is done without any modification of the passing user data by the internal network elements[1]. These "dumb and fast" networks became mature product where only speed is ever increased and there is no ambition except for simple forwarding of the data. We believe that the future-generation networks may be extended beyond that paradigm and behave as an active transport medium, which processes passing data based on data owners or data users requests. Multimedia application processing (e.g., video transcoding) and security services (data encryption over distrusted links, etc.) are a few of possible services which could be provided. The principle called "Active Networks" or "Programmable Networks" is an attempt how to build such intelligent and flexible network using current "dumb and fast" networks as an overlay network.

We can consider a computer network as a system whose end nodes provide computations up to the application level, while inner elements (routers, switches, etc.) provide computations up to the network level, and all nodes are connected via passive links. While the elements may be programmable to some extent, the control is always in the hands of network administrators. The major difference in the active network is that the elements inside the network are directly programmable by users. Also, the nodes inside the active network can provide computations up to the application level. These inner elements are called *active nodes*, *active routers*, or *programmable routers* (all three with rather identical meaning). Users and applications have the possibility of running their own programs inside the network using these active nodes as processing elements.

An application of software programmable routers in multi-user environment pose new challenges in the design of router operating systems and especially in the design of resource management system. Since more resources are shared among the users of the active router—router CPU cycles, state storage capacity, data storage together with traditional networking components like packet queues on network interfaces. To enable sharing of all these resources within the active node by its users in a secure and effective manner, much more complex Quality of Service (QoS) architecture needs to be deployed, including sophisticated resource accounting and resource scheduling algorithms that respects characteristics of individual resources.

The main goal of this paper is to propose a QoS-enabled active router (AR) architecture that supports complex QoS guaranties as described above. In order to achieve reasonable isolation among the users of the AR, the architecture is designed to facilitate implementation based on virtual machines (VM) approach [1]. The paper is organized as follows: Section II briefs previous work on a generic AR architecture, regardless of QoS, and Section III describes modified VM-based architecture suitable for QoS implementation. Proposed QoS implementation is analyzed in Section IV. Related work is summarized in Section V and concluding remarks and proposals for future work are in Section VI.

## II. Generic AN architecture

When considering the architecture of the active networks, one possible classification criterion is the way active code is delivered to the active routers [2]:

- *Active nodes* – The code of an active program is injected into the active nodes separately from the data packets. The code can be implemented either as built-in functions or during the initial phase (the opening) of the data transfer. The advantage of this architecture is that the code is injected only once and thus its size is not limited and not critical. A disadvantage lies in the necessity to inject the code before data transmission which means larger startup latency and lower flexibility as it is hard to change the code during the actual data transmission.

[1]Not including firewalls, proxies, and similar elements, where an intervention is on the one hand usually limited and on the other not user controllable.

I

- *Active packets* – Each data packet contains the program code which is extracted on an active node and executed on the data part of this packet. This approach is flexible since individual data packets in one transmission can be processed by different programs. The node needs "only" to be able to extract the code and execute it. The disadvantage is that even the limited extent of the code tends to result in a large overhead for transmitted data.
- *Active packets and active nodes* – This combination of both previous architectures allows the use of more complex programs while remaining flexible enough. Usually a program is transferred before the actual data transmission occurs, but individual data packets contain some kind of parameters or specific program commands. This supports individualized packet processing without the limitations of the active packet approach. However, the substantial initial delay (latency) is not eliminated.

For our work we use a model of active node with loadable functionality published in [3]. The proposed active network architecture uses an "active node" approach to active networking and the concept of "sessions" similar to connections in connection-oriented networks or sessions in RSVP protocol.

The structure of an active node (router) plays a key role in this model. The router is a network element which is able to accept user-supplied programs and to execute them. The processing of user code consists of two separate but communicating processes. The first process controls the session establishment and management. It has the role of a control plane in active router processing and includes a process of loading user functions into the routers along the path between the source and the destination. The functions may be either pre-loaded (before or during the connection set up) or they may be loaded on demand during the data transmission (if a new requirement arises). Bookkeeping functions are also provided by the control process. The second process performs the data packet processing which includes executing the user code.

AN model described in [3] has never been fully implemented, but main ideas from this work were successfully used for a model and implementation of user empowered UDP packet reflectors to create virtual multicasting environment as an overlay on top of current unicast networks [4]. It also served as a basis for protocol research and development, e.g., "Active Node Authentication Protocol (ANAP)" [5] and "Active Router Transport Protocol (ARTP)" [6].

## III. VM-READY AN ARCHITECTURE

Because of generic AR modular architecture, we have extended the generic AR architecture to support the complex QoS and also slightly modified the scheme in order to facilitate implementation based on virtual machines. This approach enables users not only to upload the active programs, which run inside some virtual machine, but they are allowed to upload the whole virtual machines with its operating system and let their passing data being processed by their own operating system running inside uploaded VM. VM approach ensures strict separation of different virtual machines and also allows efficient scheduling of resources to individual VMs, e.g., CPU, memory, and storage subsystem access.

The architecture of our VM-ready active router is shown in Figure 1. The bottom part is the VM-host layer where the core of the proposed VM-ready router is located. The core includes packet classifier, shared buffer pool, and packet scheduler modules. The modules relevant to resource management (resource management module and VM/AP scheduler module) are described in more detail in Section IV. Packet classifier module classifies all the incoming packets whether they belong to any active session running on the router and thus must be very efficient. It also extracts packets destined to the session management module and sends them directly to that module. The shared buffer pool module operates as the buffer space where all the incoming packets are stored before further processing and also all the outgoing packets before the packet scheduler module sends them onto the network.

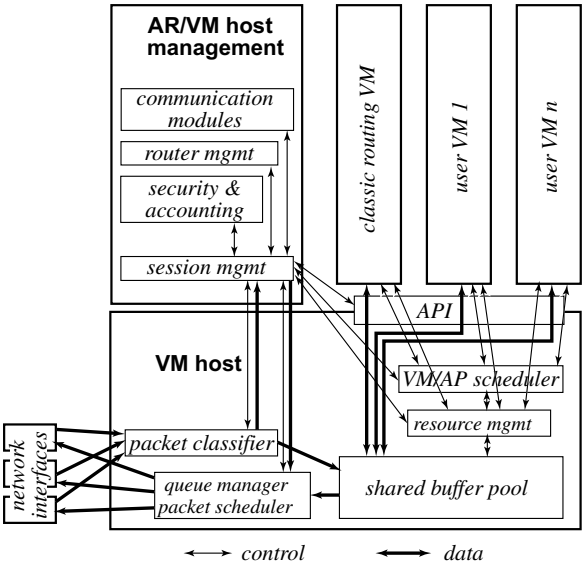



Fig. 1. VM-ready active router architecture

The VM-host management system is located in user space. Besides the other functions it has to manage the whole router functionality including uploading, starting and destroying of the virtual machines, security functions, session accounting and management. The virtual machines managed by the session management module could be either fixed, providing functionality given by system administrator, or user-loadable. The example of the fixed virtual machine could be a virtual machine providing classical routing as shown in Figure 1—it is an example of optional module, as the AR can run without the classical routing if only "active" traffic passes through the AR, e.g., if it works in a dedicated overlay network. Besides that, the one other fixed virtual machine could be started as an active program execution environment where the active programs

uploaded by users are executed. This virtual machine serves especially for backward compatibility with original generic AR and this approach does not force users to upload the whole virtual machine in the case where active program uploading is sufficient.

The VM-ready AR architecture uses a connection-oriented approach similar to the one used in active router proposed in [3]. In terms of our active architecture, the connection is also called *"(active) session"*, but each active session consists of one or more active programs/virtual machines, one or more network flows and potential QoS requirements. The association of more VMs/active programs and network flows into one session is very useful especially when creating active programs working with more than one network streams (e.g., synchronization of two RTP streams when transmitting audio and video streams separately).

Besides the other information, the session initiation request encapsulated in an active packet contains minimal resource requirements for given active session and the active router decides, whether the requirements could be satisfied. If the request could be satisfied, the session is established and all the required resources are allocated and reserved to it. Otherwise the request is refused.

Once the session is established with the required resources, the data flow through the router could be briefly described in the following way: when a packet arrives to a network interface, the packet classifier module decides, whether the incoming active packet belongs to the given AR or not, based on information from the security and accounting module. If the packet is accepted, depending on resource allocations and actual scheduling algorithm, the classifier module forwards packet to the proper VM running on the AR or the new session establishment takes place. Depending on resource management, the active packet is processed in the VM and sent into the network through the shared buffer pool.

## IV. QoS SUPPORT FOR VM-READY ACTIVE ROUTER

As obvious from VM-enabled AR architecture described above, there are the two main modules concerned with resource management: (1) resource management module and (2) VM/active program scheduler. Indirectly, the session management module also participates on this process.

**Resource management module.** This module implements the crucial resource management scheme with the following functionality:

- Possessing all the information about the resources in the AR.
- Providing necessary information to the session management module.
- Monitoring and adjusting the resources used by each active session and sending notifications to the active sessions through the session management module to inform them about the actual resource status of the AR (e.g., how many resources are available and can be used or how many resources are needed).

**VM/active program scheduler module.** This module schedules the execution of the applications and the transmission of the packets to the next node. It implements scheduling algorithms for different classes of resources to enforce the active sessions allocations of the AR resources—for the additional information on scheduling algorithms, see Section IV-B). Besides that, the accounting and resource limit checking functions are also the part of this module:

- It checks whether the active sessions are permitted to request given resources (e.g., when restricting the amount of given resources from allocating by specified users/sessions).
- It logs active sessions requests and replies from resource management system about allocating given resources (useful e.g., when active node resource utilization is paid).

### A. Resource management system

Due to the structure of active sessions where each session consists of one or more virtual machines (simply active programs) and one or more network streams, the fine-grained hierarchical design of resource allocations is very desirable. I.e., when the session possesses allocated resources, it is possible to split these resources held by the session in a way the user of the given active session wants (Figure 2).

For resource allocation and scheduling purposes, a *session element* (or just an element for short) denotes active stream ("active" network flow), virtual machine, or active program. When providing hierarchical resource management, all the schedulers must know about required resources of each element for a given active session. As said before, when creating a new active session it must request overall amount of resources wanted. If all the resources requested are available, the active session is established and all the resources are allocated. Since the active session holds an unique identifier of allocated resources, the assigned identifier must be provided when the session wants to work with a specific shared resource. This identifier is also used when the active session wants the resource management module to redistribute the allocated resources to its element(s). In this case, the (master) identifier is extended with element sub-identifier. When an element of the active session wants to use a shared resource, the master identifier or the extended identifier has to be provided depending on whether the element wants to utilize the overall amount of given resources available to the session or just the amount of resources previously redistributed inside the active session.

The resource management module thus provides mainly the following functions:

- *Create/Delete* – *Create* allows creating a resource allocation with given requirements and returns a key, a unique identifier of given allocation. *Delete* takes the key as an input parameter and removes the corresponding allocation. The allocation's resource share is then returned to the system.
- *Bind/Unbind* – *Bind* allows an active session to specify the resource requirements of elements of the active ses-

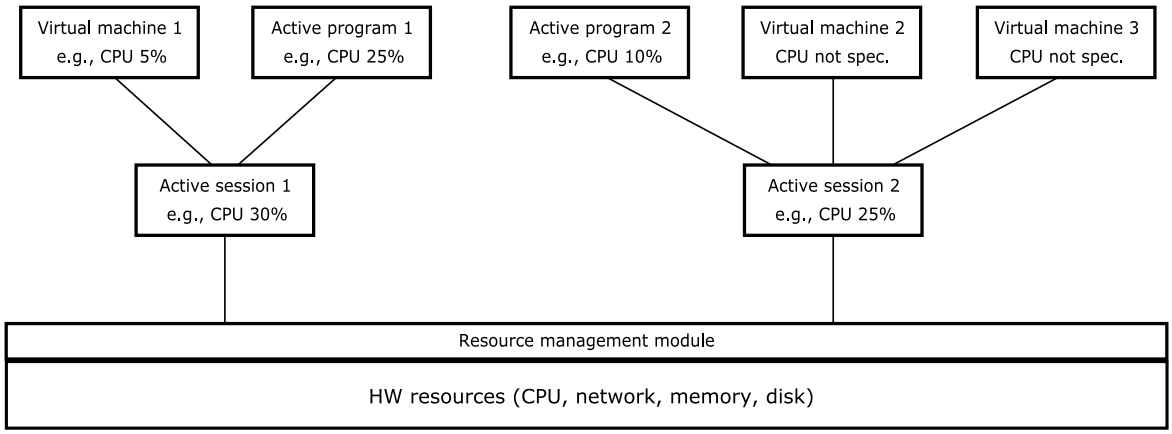



Fig. 2. Hierarchical resource assignment.

sion. Then, the session (master) key is extended to be able to unambiguously determine the active session and its element, and returned. *Unbind* deletes such binding inside the session.

- *Modify* – allows user to reconfigure a resource allocation with a given key.
- *Info* – provides the information about allocated and free resources in the system.

### B. Schedulers

Scheduling algorithms are the most important part of the whole resource management system in our active router because they affect both overall performance and keep all required resources in desired limits. Since resource characteristics vary, scheduling algorithms must be designed in a resource specific manner. For example, CPU context switching is more expensive compared to switching between flows in network scheduling [7]. Therefore, efficiency of CPU scheduling improves if active programs can receive a minimum CPU quantum before being preempted. Disk scheduling, unlike both CPU and network, must consider request locations to limit seek time and rotational latency overheads. Memory schedulers, in order to match actual memory use, must estimate the current working set of active programs. All the schedulers must therefore examine relevant resource states (e.g. disk state, whether it is spinning or parked) in addition to QoS specifications.

The resource requirements are often related to the others. For example, when requesting high network bandwidth while having only a small amount of CPU time, it is not possible to reach required bandwidth, because there is insufficient CPU time to send all the packets. Thus the scheduling algorithm's design must be sophisticated enough to take such inter-dependencies into the account.

For sake of conciseness, we do not delve into detailed description of scheduling algorithms here, but we describe the most important demands on each of the active router scheduler focusing on CPU, network, memory and disk schedulers.

Because the quality of service assurance in active routers is closely related to multimedia applications, the requirements on the scheduling algorithms in our active router are very similar to the requirements in multimedia operating systems [7].

*1) CPU scheduler:* The CPU scheduling algorithms are the best-developed scheduling algorithms in current information technology. Unfortunately, majority of proposed algorithms are QoS-unaware and thus very huge research in this area should be made.

Thanks to the hierarchical resource management system in our active router architecture the hierarchical CPU scheduling algorithm is desirable. The operating system thus partitions the CPU bandwidth among more active sessions, and each active session, in turn, partitions its allocations among its VMs or active programs.

The other desirable features of CPU scheduling algorithms are as follows:

- *Admission criteria* – the admission of a new active session should not infringe the QoS guarantees given to currently established and running active sessions. If so, necessary steps need to be taken like re-negotiation or rejecting the new active session.
- *Real-time guarantees* – the design of the CPU scheduling algorithms must satisfy real-time constraints in terms of ensuring guaranteed scheduling for each active program within their jitter bounds, if any.
- *Fairness criteria* – it should be possible to schedule all the types of active programs that are competing for the CPU—if there is non-reserved CPU time, the lower priority non-guaranteed applications should not be completely starved out of CPU by higher priority tasks corresponding to guaranteed services.
- *Maintenance and policing criteria* – policing criteria requires to ensure that the deadline violating tasks do not

infringe the QoS guarantees of other tasks competing for CPU resources. Mechanisms like software watchdog that suspends an active program on deadline violations, are means of ensuring service guarantees. The maintenance criteria imply setting up re-negotiations or dropping further requests in case of CPU overload condition.
- *Throughput criteria* – the scheduling policy should be able to schedule as many active sessions as possible.

*2) Network scheduler:* Current network scheduling algorithms are well-developed and only their adaptation to active networks is necessary. The typical objective network scheduler parameters are bandwidth, latency and jitter, and the common criteria on network schedulers are following:

- *Admission criteria* – the scheduler must ensure that the requested bandwidth plus the currently allocated bandwidth does not exceed a threshold of the total available bandwidth.
- *Real-time guarantees* – it must also ensure that the network interface scheduling delays are bounded and the enough buffer provisioning is done. The scheduling algorithm must consider that the mechanisms like retransmissions may not be suitable for applications requiring hard delay bounds.
- *Fairness criteria* – all types of applications should get a fair share of network bandwidth.
- *Maintenance and Policing criteria* – Policy criteria should ensure that the application do not take up more than the network bandwidth that has been guaranteed by QoS negotiation during active session setup.

*3) Memory scheduler:* The memory scheduling algorithms must manage the whole memory subsystem using virtualization mechanism and it must guarantee the required amount of free memory to active sessions. Because of the virtualization mechanism the appropriate allocation of free page frames and redistributing released frames to other sessions are the main jobs of the memory scheduler. The following are the requirements on the memory schedulers in order to support QoS:

- *Admission criteria* – new active session can be admitted if and only if its memory buffer requirements plus the current buffer allocations of other sessions do not exceed the threshold of the total available memory.
- *Real-time guarantees* – during the run of given active session some time-critical applications need the memory access time to be minimal. With virtual memory, it is important to have paging mechanisms that have an acceptable upper bound on access latency.
- *Fairness criteria* – the memory scheduler must ensure the minimal availability of memory buffers for all the active sessions and their elements.
- *Maintenance and Policing criteria* – maintenance criteria require setting up re-negotiations or dropping further requests in case of a buffer shortage. Policy criteria may require that the offending active session should be notified for the re-negotiations or in the extreme case terminated.

*4) Disk (I/O) scheduler:* While a secondary data storage is not a traditional router resource, it is very important for the active router with QoS support. The disk scheduler may support either the transfer bandwidth of given disk or the amount of free disk space only or both. The requirements on suitable disk scheduling algorithms could be summarized into the following criteria:

- *Admission criteria* – the effective disk transfer bandwidth is reduced due to seeking and latency overheads, which are a function of the disk scheduling algorithm and the disk request size. The admission criterion ensures that the sum of the data rates of all the active sessions, including the new one, do not exceed the effective disk transfer bandwidth.
- *Real-time guarantees* – the scheduler must be able to schedule the disk accesses for all admitted streams so as to meet their data rate guarantees and the response time for all the streams must be acceptable.
- *Fairness criteria* – provisioning may be done to ensure that all types of active sessions get a fair share of disk transfer bandwidth.
- *Maintenance criteria* – the scheduling algorithm has to monitor the data rates being provided to real-time streams with respect to the guarantees provided before; the QoS re-negotiations must be provided in cases of shortfalls.
- *Resource reservation* – except the reservations of disk bandwidth each active session requires at a minimum a buffer for the consuming virtual machine (resp. active program) and a buffer for the producing virtual machine (active program). Thus, this amount of memory needs to be reserved for each admitted stream.

## V. Related Work

In this section we brief the results of our work in the context of related projects in the resource management and QoS assurance in active networks research. We also notice relevant projects whose ideas may be implemented in our AR architecture.

An active node architecture with resource management [8] is an attempt how to introduce the architecture with explicit resource management system, which also provides an adaptation among different applications. The node operating system is based on the Janos project [9] developed at the university of Utah. The Janos project is in comparison with our work oriented to the execution of untrusted Java byte-code only and thus has limited flexibility. As a part of this project the method for the description of resource requirements from applications using the resource vectors and resource vectors space were introduced. We will explore the resource vector method in more detail and assess its possible application to our router architecture.

The CROSS project [10] is another attempt of introducing the resource management system in software-programmable router operating systems. This project was proposed by David K. Y. Yau and Xiangjing Chen in 2001 and it uses virtual machines as the active programs providing thus fixed

router functionality only, but higher security and efficiency, because the CROSS system communicates directly with the hardware layer.

Friendly virtual machines [11] devises techniques that enable multiple virtual machines to share underlying resources on the same host both fairly and effectively. Instead of deploying complex resource management techniques in the hosting infrastructure, an alternative approach of self-adaptation in the virtual machines themselves was introduced based on feedback about resource usage and availability. Thus, the virtual machines that adjust their demands for system resources, so they are both efficiently and fairly allocated to competing virtual machines, are very important idea for the resource management subsystem in our router, where the principle of Friendly virtual machines could be used for the "competing" virtual machines with no explicit resource requirements.

QoS specification languages for distributed multimedia applications could be used for the description and negotiation of QoS requirements of active sessions in our architecture. Such languages were studied by Jingwen Jin and Klara Nahrstedt in [12]. They studied lots of languages including script languages and XML-based markup languages. The languages for the hierarchical resource requirements description probably suitable for our router architecture were also studied.

## VI. Conclusions And Future Work

In this paper, we have proposed a virtual machine oriented active router architecture and studied the resource requirements and QoS implementation. The typical scenario of hierarchical resource management system has been explored and the requirements on the schedulers for such QoS-enabled active router, concerning the CPU scheduler, network scheduler, memory scheduler, and disk scheduler were also discussed.

The main feature of our router architecture is that it provides a predictable and assured access for active sessions to system resources. These resources can be subsequently redistributed to multiple virtual machines, giving flexible choice and the ability for services to seamlessly evolve.

Concerning the future challenges, the proposed router architecture will be implemented based on Xen virtual machine monitor [13]. Further we want to explore extending current architecture into a distributed environment to be able to deal with high-speed networks. In this case, all the resource schedulers, the whole router, and session management systems must be modified. For the efficiency purposes, another interesting topic for our future work is the implementation of some parts of the router architecture (especially the packet classifier module) in hardware, e.g., based on FPGA-based programmable hardware cards [14].

## Acknowledgments

This project has been supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201), "Parallel and Distributed Systems" (MŠM 0021622419), and "Integrated Approach to Education of PhD Students in the Area of Parallel and Distributed Systems" (No. 102/05/H050).

## References

[1] J. E. Smith and R. Nair, *Virtual Machines: Versatile Platforms for Systems and Processes*. Elsevier Inc., 2005.

[2] K. Psounis, "Active networks: Applications, security, safety and architectures," *IEEE Communication Surveys*, 1999.

[3] E. Hladká and Z. Salvet, "An active network architecture: Distributed computer or transport medium," in *Networking – ICN 2001: First International Conference Colmar, France, July 9-13, 2001, Proceedings, Part II*, ser. Lecture Notes in Computer Science, P. Lorenz, Ed., vol. 2094. Heidelberg: Springer-Verlag, Jan. 2001, pp. 612–619.

[4] E. Hladká, P. Holub, and J. Denemark, "An active network architecture: Distributed computer or transport medium," in *3rd International Conference on Networking (ICN'04)*, Gosier, Guadeloupe, Mar. 2004, pp. 338–343.

[5] J. Denemark, "Autentizace v aktivních sítích (authentication in active networks)," Master's thesis, Faculty of Informatics, Masaryk University in Brno, Apr. 2003, czech only.

[6] T. Rebok, "Active router communication layer," CESNET, Tech. Rep. 11/2004, 2004. [Online]. Available: http://www.cesnet.cz/doc/techzpravy/2004/artp-protocol/

[7] B. Ghose, V. Jain, and V. Gopal, "Characterizing qos-awareness in multimedia operating systems," 1999, http://computing.breinestorm.net/qos+cpu+scheduling+criteria+admission/%.

[8] Y. Li and L. Wolf, "An active network node system with adaptive resource management," in *International Conference on Telecommunications*, June 2002.

[9] P. Tullmann, M. Hibler, and J. Lepreau, "Janos: A java-oriented os for active network nodes," 2001. [Online]. Available: citeseer.ist.psu.edu/652534.html

[10] D. K. Y. Yau and X. Chen, "Resource management in software programmable router operating systems," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 3, Mar. 2001. [Online]. Available: http://citeseer.ist.psu.edu/324757.html

[11] Y. Zhang, A. Bestavros, M. Guirguis, I. Matta, and R. West, "Friendly virtual machines - leveraging a feedback-control model for application adaptation." [Online]. Available: citeseer.ist.psu.edu/article/zhang04friendly.html

[12] J. Jin and K. Nahrstedt, "Qos specification languages for distributed multimedia applications: A survey and taxonomy," *IEEE MultiMedia*, vol. 11, no. 3, pp. 74–87, 2004.

[13] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. B. m, and R. Neugebauer, "Xen and the Art of Virtualization," in *Proceedings of the ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, USA, Oct. 2003.

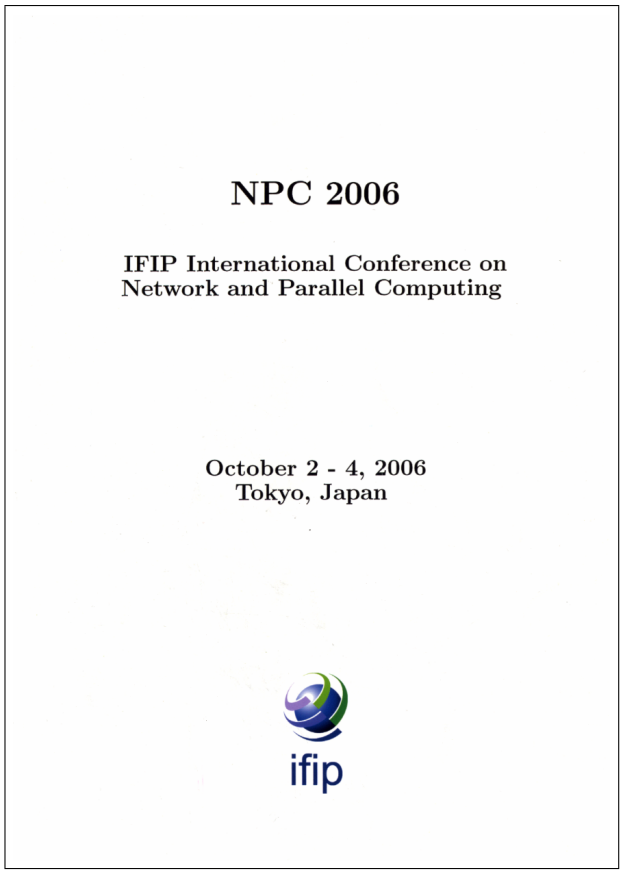
[14] J. Novotný, O. Fučík, and D. Antoš, "Project of IPv6 Router with FPGA Hardware Accelerator," in *Field-Programmable Logic and Applications, 13th International Conference FPL 2003*, P. Y. Cheung, G. A. Constantinides, and J. T. de Sousa, Eds., vol. 2778. Springer Verlag, September 2003, pp. 964–967.

# Appendix J

# Distributed Active Element for High-Performance Data Distribution

by Petr Holub and Eva Hladká

IFIP International Conference on Network and Parallel Computing, NPC 2006, Tokio, Japan, October 2006. Proceedings. pp. 27–36, 10 p.



# Distributed Active Element for High-Performance Data Distribution

Petr Holub[1*] Eva Hladká[2†]

[1]*Institute of Computer Science*
*Masaryk University,*
*Botanická 68a, 602 00 Brno, Czech Republic*
[2]*Faculty of Informatics,*
*Masaryk University,*
*Botanická 68a, 602 00 Brno, Czech Republic*

## 1 Introduction

Requirement on multi-point data distribution in IP networks assumes some distribution service, be it implemented as a part of network native services (IP multicast) or user-empowered solution (data reflectors). In our previous work, we have introduced user-empowered distribution networks based on Active Elements (AE) [1], which scale well in terms of number of clients connected. It is however not sufficient in terms of scalability with respect to bandwidth of each single stream distributed, i. e. it is not suitable for distributing streams whose bandwidth exceeds capacity of each single AE [2].

In order to improves the scalability with respect to the bandwidth of a stream, we propose a concept of distributed AE, suitable for implementing on computer clusters with low-latency internal interconnection. Its architecture is based on parallelizing whole AE architecture including listener and sender modules. which however brings problems with packet reordering in sending part. While packet reordering is largely unwanted for general network element (e.g. router) behavior, as it severely tampers performance especially of TCP-based applications, it is more acceptable for multimedia application that rely on UDP protocol and thus need to handle potential packet reordering anyway. Simple solution is a design with all distributed modules except for sender module—while this would help for computationally intensive operations on the streams, it is of little use for high-bandwidth streams. In this paper we show, that even when having multiple sending modules with no explicit synchronization, the reordering introduced by the distributed AE has an upper bound for real-time synchronous applications under certain assumptions. It can be further reduced by implementing proposed Fast Circulating Token protocol.

**Related work.** Distribution of multimedia data over IP network leads to a multicast schema. However, as the native multicast solution is not always reliable or even available, other distribution schemes were developed following approach of multicast virtualization. They are usually based on a central distribution unit—a reflector—like the H.323 MCUs or reflectors provided in the Virtual Room Videoconferencing System (VRVS)[1]. The successor of VRVS called Enabling Virtual Organizations (EVO)[3] is based on self-organization of system of reflectors, again not empowering the end-user with tools to change the distribution topology. Other simpler UDP packet reflectors include rcbridge [4], reflector[2], and Alkit Reflex[3].

Another area related to this paper is utilization of computer clusters as either distributed routers or distributed servers. Project Suez [5, 6] is a distributed router based on commodity PC cluster with Myrinet interconnection with each node of the cluster having one internal interface to the Myrinet switch and optionally one or more external interfaces. Suez uses a routing-table search algorithm that exploits CPU cache for fast lookup by treating IP addresses directly as virtual addresses. Another project which distributes processing load on active network elements is Active Network Node [7] that relies on specialized hardware. Software DSM project [8] attempts to build efficient distributed memory for the closely coupled clusters for using them as active routers. There is yet another similar project called Cluster-based Active Network Router [9]. However none of the above mentioned projects addresses finer than per-address network load distribution and thus there is no need for solving packet reordering issues.

There is a number of distributed servers based on employing computer clusters. Most distributed servers are prototyped as web servers [10, 11, 12] for simplicity reasons and because rather standard and straightforward performance evaluation is available. For example, Carrera and Bianchini recently demonstrated cluster based web server called PRESS [13] concentrating on demonstrating advantages of user level communication like low processor overhead, remote memory accesses, and zero-copy transfers.

**Paper organization.** This paper is organized as follows. Section 2 discusses general architecture of the distributed AE and its behavior and the packet reordering

*hopet@ics.muni.cz
†eva@fi.muni.cz

[1] http://www.vrvs.org/
[2] http://www.cs.ucl.ac.uk/staff/s.bhatti/teaching/z02/reflector.html
[3] http://w2.alkit.se/reflex/

J

from the theoretical point of view. Prototype implementation is described and evaluated in Section 3. The paper concludes with future work and concluding remarks in Section 4.

## 2 Distributed Active Element

We are proposing the distributed AE based on architecture of AE described in [1] and it is partly determined by requirement of implementability on existing tightly coupled clusters with low latency interconnection. The distributed AE implementation assumes the infrastructure as shown in Figure 1. The computing nodes form a computer cluster with each node having two connections: (1) *low-latency control connection* used for internal communication and synchronization inside the distributed AE, and (2) *data connection* used for receiving and sending the data. Low latency interconnection is necessary since current common network interfaces like Gigabit Ethernet provide large bandwidth, but latency of the transmission is still in order of hundreds of $\mu$s, which is not suitable for fast synchronization. Specialized low-latency interconnects like Myrinet provide as low latency as 6 $\mu$s, which is comparable to message passing between threads on a single computer.

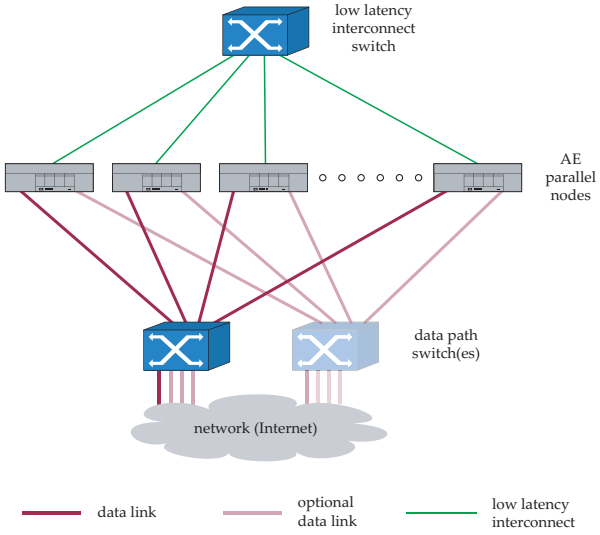



Figure 1: Model infrastructure for implementing the distributed AE.

The incoming data needs to be first distributed across the multiple parallel units of the distributed AE, processed in these units, and finally aggregated and send over the network to the listening clients. Thus the architecture comprises three major parts:

- *Distribution unit* takes care of ingress data flow distribution over multiple parallel distributed AE units. When the distribution unit is part of the same L2 domain as parallel AE units, it may operate on L2 addresses only (e. g. Ethernet addresses in operation similar to VRRP[4] or CARP[5] protocols), otherwise L3 (usually IP) addressing is needed.

- *Parallel AE unit* is a complete instance of AE with modified sender module to allow for possible synchronization. It has the kernel with administrative submodules, session management, processor schedulers and AAA submodules. Data are received using network listener modules, stored into shared memory (shared across the instance of the reflector only, not across multiple AE unit instances), processed by zero or more processors, distribution lists filled up with either one of processors or with session management and finally sent with the sender module. Unless there is some complex data processing involved, data passes through distributed AE unit in zero copy mode for performance reasons.

  The network management module handles communication with distribution unit and also communication with other distributed AE units if AE ring is to be set up and maintained for the Fast Circulating Token protocol (Section 2.2). However, handling token itself for this protocol is performed directly by the sender module in order to minimize operation overhead.

- *Aggregation unit* aggregates the resulting traffic to output network line(s). Because the AE element is often used as data multiplication unit, we assume that output data flow from the distributed AE is larger than input data flow. Thus we need a unit that is even more powerful that the input load distribution unit and in most cases, cheap custom made software implementation is not available and we have to use available hardware solution like aggregating switch. However, in that case we must not assume any further behavior of the aggregating unit except for two things: first, it is over-provisioned enough not to loose any data and second, it has limited buffer space available.

Whole architecture supports user-empowered operation, there is still no need for running any part of it in kernel space. The only administrative requirement is that the cluster environment needs to be set up together with networking infrastructure including distribution and aggregation units.

In order to set up and maintain the distributed AE, some protocol is needed – it has been described in [14] in detail and it is out of scope of this paper. In the rest of the paper we describe operation of distributed AE in static environment, where there is constant number of AEs participating in distributed processing and the AEs work reliably.

In order to evaluate our models theoretically, we need to introduce an idealized environment:

[4] http://www.ietf.org/html.charters/vrrp-charter.html
[5] http://www.countersiege.com/doc/pfsync-carp/

J

- The *ideal network* is a network in which no data are lost, corrupted, nor reordered. It also provides instant delivery, i. e. it introduces zero latency.
- The *ideal multimedia traffic* has bandwidth $b$ and independent packets of exactly same size $s^p$, which is equal or smaller than MTU of the underlying network. The packets are sent in regular intervals. All the queue sized below are expressed in units of packet size $s_p$. In order to isolate reordering introduced by the distributed AE, we assume that the ideal multimedia traffic has no reordering prior to entering distribution unit.
- The *ideal aggregating unit* has $n$ input interfaces with the same parameters and one output interface with capacity equal or bigger than the $n$ inputs together. It reads packets from the size-limited input interfaces queues and sends them on output interface in such a way, that packets are never lost. The speed is $b_j^{SW}$ for $j$-th input interface and each input queue has equal size of $s_i^{SW}$ for each input interface. In order not to lose any input data, the ideal aggregating unit needs to fulfill the following requirement in the steady state: $\sum_j b_j^{SW} \leq b_o^{SW}$.
- The *ideal AE* has processing capacity equal or higher than stream bandwidth and it has an input queue size of $q_i^{AE}$. All the parallel units of the ideal AE have the same parameters and performance and the total bandwidth of the traffic is divided into streams with the same parameters. The ideal AE introduces no losses, nor data corruption, nor data reordering in the data stream.

### 2.1 Ingress Distribution

The ingress data distribution takes care of distributing incoming data across different paths inside the distributed AE. For the ideal distributed AE the ideal distribution unit distributes packets in round-robin fashion. In each *round*, it distributes $n$ packets, one to each of the parallel units. The distribution unit marks round number into each packet.

Such an ideal distribution might not be suitable in the cases, when parallel AE units are of unequal performance or when data stream packets are not independent and the processing needs to have all the inter-dependent packets through the same path. When parallel AE units do not have the same performance, the load balancer can send multiple packets in each round to the same parallel path. All the packets sent in one round are marked with the same round number. Sample packet distribution is shown in Figure 3.

### 2.2 Egress Synchronization

#### 2.2.1 No explicit synchronization.

The simplest model for egress synchronization is to use no synchronization at all. However, with this model and limited buffers on the input interfaces of the AEs, there is still some implicit synchronization achieved.

It can be shown the maximum reordering induced by an ideal distributed AE with no explicit egress synchronization and ideal aggregating unit is

$$n(s_i^{AE} + s_o^{AE} + s_i^{SW} + 1),$$

where $n$ is the number of parallel AE units when all queues operate in FIFO tail-drop mode. Detailed proof can be found in [14].

#### 2.2.2 Fast Circulating Token.

In order to decrease packet reordering introduced by the distributed AE, we propose a distributed algorithm for achieving less packet reordering compared to no explicit synchronization. The nodes are ordered in a ring with one node elected as a master node and they circulate a token which serves as a barrier so that no node can run too much ahead with sending data. The mechanism is called *Fast Circulating Token* (FCT) since the token is not held for the entire time period of data sending as usual in the token ring networks.

Because of real world implementation of data packet sending in common operating systems, we assume, that sending procedure is non-preemptive, i. e. once a packet is being sent, this process can be interrupted after sending is finished. Further we assume that token reception event processing has precedence over any other event processing in the distributed AE. If there are multiple token events waiting, they are processed in FIFO way. However, as the data sending is non-preemptive, if the token arrives in the middle of data packet sending, it will be handled just after that packet sending is finished.

The token carries the following information: (1) *round number* – corresponds to round number from distribution unit, which is set and incremented on master node, (2) *last round-trip time*, and (3) *holding time left after traveling from master to current node*

Depending on implementation circumstances, `timeLeft()` function may be used to allow keeping token on other nodes than master for limited amount of time. This might be needed if e. g. the master node is considerably faster than other nodes. For ideal distributed AE, `timeLeft()` returns 0.

After more detailed analysis [14], it can be shown the maximum reordering induced by an ideal distributed AE with FCT egress synchronization and ideal aggregating unit is

$$n(s_i^{SW} + 3),$$

where $n$ is the number of parallel AE units when all queues operate in FIFO tail-drop mode.

When operating in a non-ideal environment, there are several complications that needs to be taken into account:

- packet reordering, either before data reach distributed AE, or on a single parallel path inside distributed AE – possible implementation of the first condition after token reception influences whether excessive packet reordering will be converted to packet loss or not,
- due to unequal performance of parallel paths, load balancing may be deployed – again the reordering

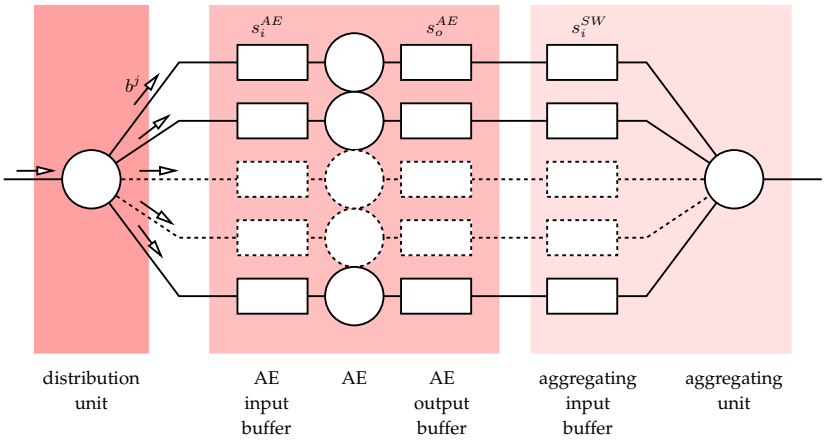



Figure 2: Model of the ideal distributed AE with ideal aggregation unit.

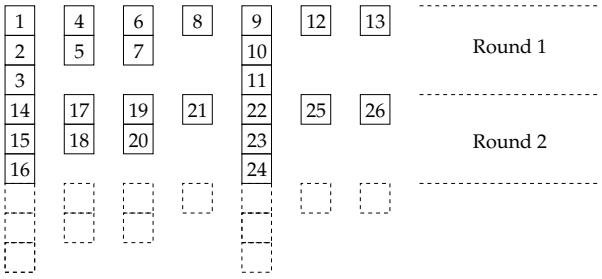



Figure 3: Sample load balancing packet distribution for distributed AE.

.

of two consecutive packet is limited by size of two consecutive rounds, but each round may have more than $n$ packets depending on load balancing scheme used.

#### 2.2.3 Exact Order Sending.

It is possible to design sending protocol that results into exact ordering, but it requires defined behavior of aggregation unit and thus it is not suitable for implementation on commodity hardware like aggregating switches.

One possibility is that aggregation unit behaves similar to sending modules with FCT protocol, i. e. it reads packets from input either in the same round robin way distribution unit distributes packets and utilizes mark of round number in each packet to recognize when the packet is ready to be sent. When each parallel path is ideal, namely there is no packet loss or corruption, even packet round numbering might not be necessary. However, in order to protect aggregation against lost and/or corrupted packets which would make one of the queues go ahead of the rest, it is advisable to stick to round numbering of the packets. For such operation, an alternative packet distribution shown in Figure 5 is more appropriate—compared to the distribution shown in Figure 3, it has smaller rounds containing either zero or one packet. Instead of sending no packet, it actually needs to send empty round marker to let the aggregator know that there is no need to wait for the packet to arrive. It not efficient when circulating token is present as there are more rounds and thus it pronounces overhead of the token.

Such protocol might be implementable on custom hardware, e. g. data switch, or custom network processor, or at least some programmable routing device like FPGA-enabled routing cards.

## 3 Prototype Performance Evaluation

Prototype implementation of the distributed AE is implemented in ANSI C language for portability and performance reasons. The implementation comprises two parts: a load distribution library and distributed AE itself.

Because of lack of flexible enough load distribution hardware unit, we have implemented it as a library, which allows simple replacement of standard UDP related sending functions in existing applications and allows developers to have defined type of load distribution—either pure round robin or load balancing.

Each parallel AE element uses threaded modular implementation based on architecture described in Section 2. Internal buffering capacity of each AE node has been set to 500 packets. Explicit synchronization using

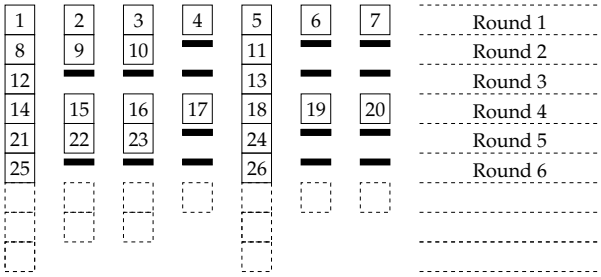



Figure 5: Alternative load balancing packet distribution. Empty round markers are shown as small black filled rectangles.

```
1  rnd := 0;
2  finish := false;
3  while ¬finish do
4        if ¬(master ∧ rnd = 0)
5          token_ready := 0;
6          do
7              if test_recv_token(rnd_no, last_RTT, t_left)
8                token_ready := 1;
9                if ¬master ∧ t_left = 0
10                 pass_token(rnd_no, last_RTT, t_left);
11               fi
12             fi
13             if test_recv_finish()
14               finish := true;
15             fi
16             send_packet(rnd);
17         while ¬(token_ready ∨ finish) od
18         if finish
19           break;
20         fi
21       fi
22       if master
23         send_all_packets(rnd);
24         rnd := get_rnd_from_queue();
25         last_RTT := updateRTT();
26         t_left := timeLeft();
27         pass_token(rnd, last_RTT, t_left);
28       else
29             while t_left > 0 ∧ is_packet(rnd) > 1 do
30                   send_packet(rnd);
31                   t_left := t_left − 1;
32             od
33             pass_token(rnd_no, last_RTT, t_left);
34             discard_packets(rnd);
35             rnd := rnd_no;
36       fi
37       if master ∧ finish_requested
38         foreach s ∈ slaves do
39           send_finish(s);
40         od
41         finish := true;
42       fi
43 od
```

Figure 4: Fast Circulating Token algorithm.

FCT protocol has been implemented using MPICH implementation[6] of MPI[7] built with low-latency Myrinet GM 2.0 API[8] (so called MPICH–GM). Prototype implementation has been tested and known to work on Linux.

For cost-effective prototype implementation, the aggregation unit was a implemented as commodity switch satisfying condition that egress link capacity is equal or larger than ingress capacities and with sufficient capacity of internal switching matrix.

### 3.1 Experimental Setup

In order to evaluate performance and behavior of the distribute AE experimentally, we have set up a testbed shown in Figure 6 comprising eight machines and two switches:

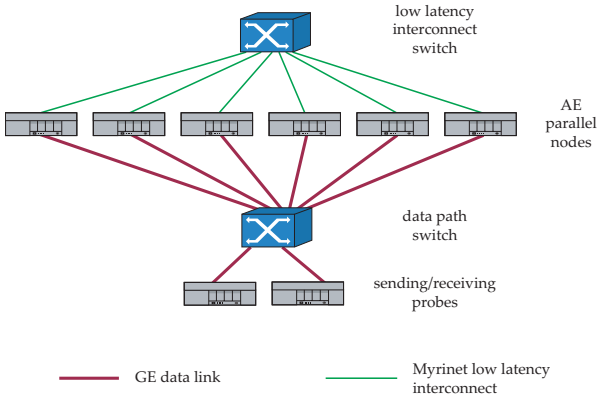



Figure 6: Distributed AE testbed setup.

- A data switch HP ProCurve 6108 with 8× Gigabit Ethernet full wire-speed ports. Manufacturer-specified switching capacity is 16 Gbps and switching performance of 11.9 million packets per second is given for 64 B packets.
- A low-latency Myrinet M3-E32 switch with M3–SW16–8F interface cards that created the control

[6] http://www-unix.mcs.anl.gov/mpi/mpich/
[7] http://www.mpi-forum.org/
[8] http://www.myri.com/scs/GM-2/doc/html/

plane for passing control information like token for FCT protocol. According to manufacturer's specifications and benchmarks, it features as low one-way latency as 6.3 $\mu$s for short messages up to approximately 100 B with MPI and GM-2.0 API[9].

- 6× PCs used as the parallel nodes for the distributed AE with configuration shown in Table 1. Each node was connected to via data link to HP switch via full-duplex Gigabit Ethernet and also to Myrinet switch for control information passing.
- Sender and receiver PCs with the same configuration (Table 1), that have been used for generating traffic and collecting and analyzing results. Both computers were connected to the HP switch via full-duplex Gigabit-Ethernet.

| | *Configuration* |
|---|---|
| Brand | HP ProLiant |
| Model | DL 360 G3 |
| Processor | 2× Intel Xeon 2.40 GHz |
| Front-side bus | 533 MHz |
| Memory | 2 GB (PC 2100 DDR) |
| GE NIC | 2× Broadcom Corporation NetXtreme BCM5703 (rev. 2) |
| Myrinet NIC | M3F-PCI64C-2 |
| Operating system | Linux Debian Woody kernel 2.4.29 SMP |
| GM Socket version | 2.0.8 |

Table 1: Configuration of distributed AE nodes and sending/receiving probes.

The data flows were generated using simple RTP-compliant sending application and data reception was done by receiver, which also computed all the required statistics. We have evaluated it with sending data stream with bandwidth up to 1 Gbps and measurements above 1 Gbps will be carried out in the near future.

### 3.2 Performance Evaluation

Performance of the distributed AE prototype without explicit sending synchronization and with FCT-based synchronization is shown in Figures 7 and 8 respectively.

It turns out that single path AE (equivalent to single reflector) is not capable of processing streams beyond 600 Mbps on given testbed infrastructure without packet loss. This is in accordance with findings in [15], where stand-alone centralistic reflector was examined on testbed infrastructure with even slightly less performance.

Contrary to the stand-alone reflector, the distributed AE prototype can process and distribute streams up to 1 Gbps using 1472 B UDP payload without significant packet loss starting with two parallel paths. Jitter, usually explained as delay dispersion, is calculated according

[9] `http://www.myri.com/myrinet/performance/#GM-2.0`

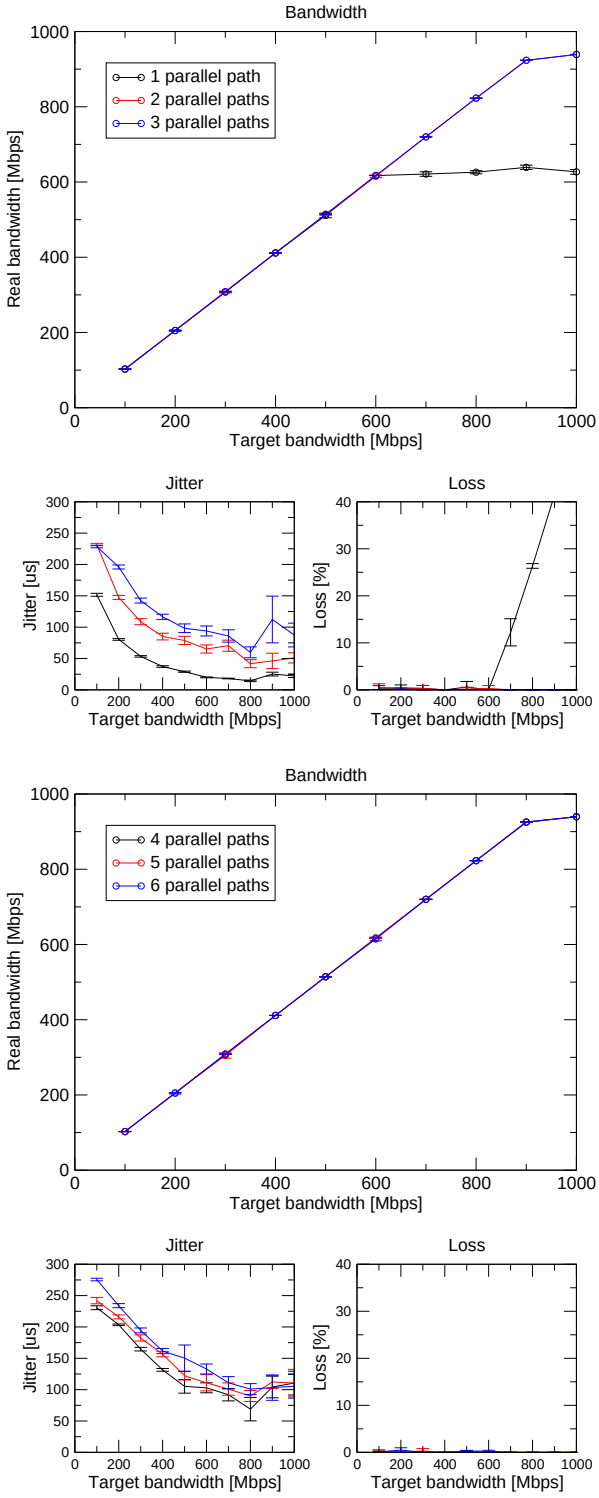



Figure 7: Forwarding performance of distributed AE without explicit synchronization for number of paths 1 through 6.

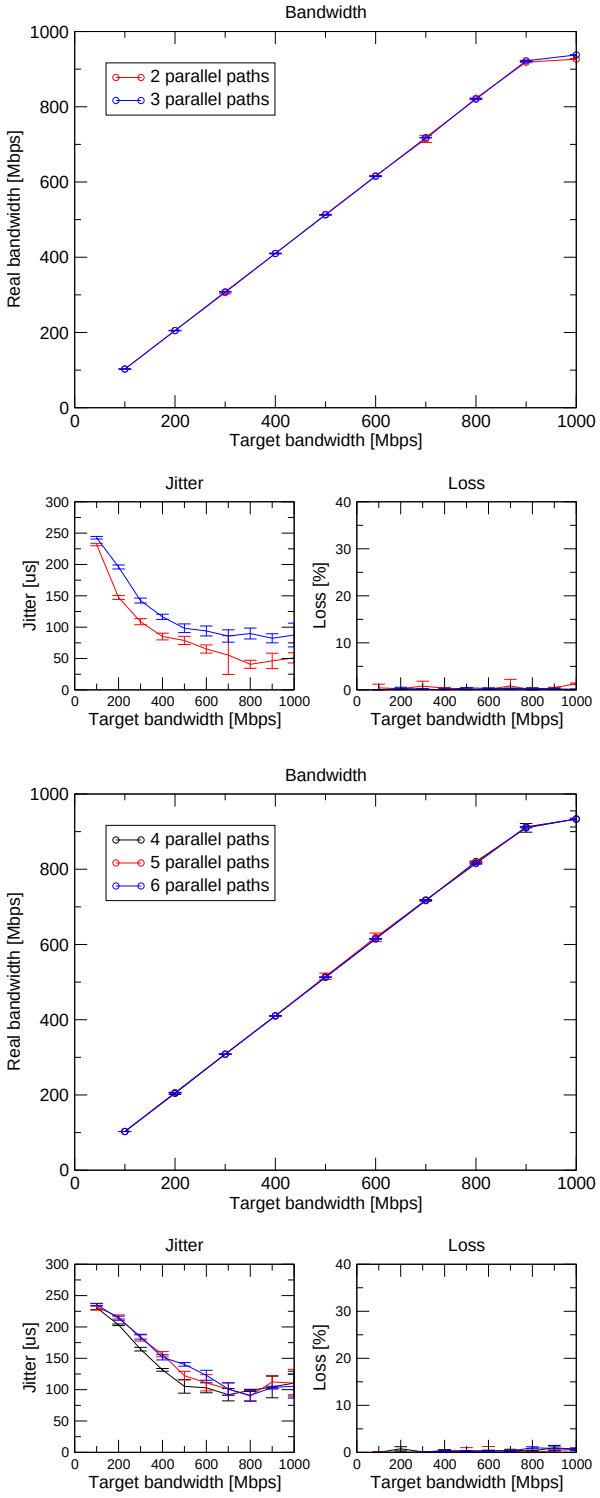



Figure 8: Forwarding performance of distributed AE with synchronization using FCT for number of paths 2 through 6.

to RFC 3551 based on arrivals of two consecutive packets as

$$D_{i,j} = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i),$$

$$J = J + \frac{1}{16}(|D_{i-1,i}| - J),$$

where $S_i$ is the RTP timestamp from packet $i$, and $R_i$ is the time of arrival in RTP timestamp units for packet $i$ for two consecutive packets $i$ and $j$. From this definition it doesn't include packet reordering discussed below and it only measures "evenness" of packet arrivals independent of packet order. It starts about 300 $\mu$s when sending 100 Mbps and slowly drops to below 100 $\mu$s as the data rate increases. For more parallel paths, it slightly raises and this effect is more pronounced on distributed AE without egress synchronization.

### 3.3 Packet Loss and Reordering Evaluation

We have measured and analyzed also packet reordering in order to experimentally compare behavior of distributed AE without explicit egress synchronization and distributed AE with synchronization using FCT protocol. The detailed reordering samples can be found in [14].

The reordering is expressed as the difference between sequence numbers of two consecutive packets. Thus, if all the sequentially numbered packets arrive in the same order they were sent, all the differences are +1. Higher number than +1 means, that some packets were skipped forth (either because of packet reordering or because of packet loss) while negative number means stepping back in packet numbering (due to packet reordering only). Value of 0 occurs when duplicate packets arrive immediately following each other. $\min\{j\}$ is the maximum negative difference in sequence numbers of successively received packets and $\max\{j\}$ is the maximum positive difference.

For any interval of arrivals of two or more packets, the following equation holds

$$\underbrace{\sum_{j=\min\{j\}}^{-1} jh_j}_{H^-} + \underbrace{h_1}_{H^1} + \underbrace{\sum_{j=2}^{\max\{j\}} jh_j}_{H^+} = \Delta, \qquad (1)$$

where $\Delta$ is difference between sequence number of last and first packet in the observed interval. Also, for the any interval of arrivals of more than one packet, the following equation holds:

$$\Pi + \underbrace{\sum_{j=\min\{j\}}^{-1} h_j}_{N^-} + \underbrace{h_1}_{N^1} + \underbrace{\sum_{j=2}^{\max\{j\}} h_j}_{N^+} - \delta = \Delta. \qquad (2)$$

where $\Pi$ is number of lost packets and $\delta$ is a number of duplicated packets that are not included in $h_0$. Proofs for both can be found in [14]. By combining both equations (1) and (2), we can derive packet loss as $\Pi = H^- + H^+ - N^+ - N^- + \delta$. Because positive part of the graph described by $H^+$ or $N^+$ includes also packet loss, the negative part of the graph described by $H^-$ or $N^-$ can be seen as measure of packet reordering.

| *With no egress synchronization* | | | | *With FCT protocol* | | | |
|---|---|---|---|---|---|---|---|
| *2 parallel paths* | | | | | | | |
| $BW$ [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ | $BW$ [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ |
| 100 | -15 | -58 | 18 | 100 | -1 | -19 | 19 |
| 200 | -7 | -35 | 11 | 200 | -3 | -242 | 184 |
| 300 | -49 | -339 | 55 | 300 | -3 | -502 | 466 |
| 400 | -13 | -194 | 150 | 400 | -3 | -4768 | 4724 |
| 500 | -15 | -1370 | 1266 | 500 | -3 | -4563 | 4533 |
| 600 | -19 | -12240 | 11726 | 600 | -3 | -36270 | 36220 |
| 700 | -25 | -48405 | 47871 | 700 | -3 | -109264 | 109238 |
| 800 | -33 | -105801 | 103793 | 800 | -3 | -176277 | 176269 |
| 900 | -35 | -239722 | 234334 | 900 | -3 | -98721 | 98703 |
| 1000 | -35 | -265286 | 258558 | 1000 | -3 | -55548 | 55476 |
| *3 parallel paths* | | | | | | | |
| $BW$ [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ | $BW$ [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ |
| 100 | -13 | -37 | 11 | 100 | -2 | -16 | 13 |
| 200 | -32 | -149 | 20 | 200 | -5 | -269 | 154 |
| 300 | -376 | -28171 | 1230 | 300 | -5 | -606 | 432 |
| 400 | -113 | -2452 | 316 | 400 | -5 | -1357 | 976 |
| 500 | -17 | -576 | 474 | 500 | -5 | -2169 | 1897 |
| 600 | -104 | -3759 | 1473 | 600 | -5 | -5688 | 5268 |
| 700 | -154 | -9608 | 4129 | 700 | -5 | -9180 | 8303 |
| 800 | -26 | -10147 | 8821 | 800 | -5 | -14522 | 13069 |
| 900 | -32 | -25210 | 20327 | 900 | -5 | -32276 | 28228 |
| 1000 | -32 | -31032 | 24806 | 1000 | -5 | -29819 | 25872 |

Table 2: Comparison of reordering for distributed AEs with no explicit egress synchronization and with synchronization using FCT. Part 1.

The difference between the $H$-sums and the $N$-sums is that the $H$-sums are "weighted sums". Thus the more packets are farther from 1 in either direction, the higher the absolute value of $H$-sums are, while the $N$-sums remain the same. All the terms in the $N^-$, $N^+$, and $H^+$ are positive and all the terms in the $H^-$ are negative. If $H^- \approx N^-$, the vast majority of out-of-order packets in the negative part is reordered by $j = -1$.

**Stand-alone reflector.** As discussed above, the stand-alone reflector is not capable of forwarding data streams above 600 Mbps without packet loss, where only +1 reordering value is only populated up to 600 Mbps and asymmetrical distribution leaning toward positive values is shown for 700 Mbps and above. The bigger the loss is, the larger the sum $H^+$ is. Because no reordering nor duplicates are introduced either, $H^- = N^- = h_0 = 0$.

**Distributed AE without explicit synchronization.** While the distributed AE performs very well in terms of low packet loss, it introduces severe packet reordering in both terms of maximum reordering (expressed as the minimum reordering populated in histogram, i. e. $\min\{j\}$) and also numbers of packets reordered (expressed by $H^-$ and $N^-$), as obvious from left column of Tables 2 through 4. Furthermore, the reordering fluctuates very significantly in time and is hardly reproducible.

**Distributed AE with synchronization using FCT protocol.** Compared to distributed AE without explicit synchronization, the FCT protocol allows distributed AE to work in much more predictable manner. It reduces packet reordering to just a very few packets and it also reduces reordering both in terms of maximum reordering ($\min\{j\}$) and numbers of packets reordered ($H^-$ and $N^-$), as can be seen from right column of Tables 2 through 4. The maximum reordering grows $\min\{j\} = 2n+1$ where $n$ is number of parallel AE paths, which indicates that the switch in the testbed works in rather very precise round robin fashion when aggregating flows from multiple interfaces. The number of reordered packets is comparable for lower number of parallel paths for both with and without synchronization and as the number of parallel path grows, the synchronized version becomes better more than 3× for higher bandwidths. The results are also in agreement with the theoretical analysis of *maximum* reordering, as $n(s_i^{SW}+3) > 2n-1$.

Detailed reordering histograms for both distributed AE without explicit synchronization and with it can be found in [14].

Token round-trip time has been also periodically sampled[10] and it ranges between 14 $\mu$s for 2 parallel AE paths

[10] In order not to influence results of measurements of distributed AE performance, it was not possible to continuously gather individual token round-trip times. Thus only sample values were periodically gathered.

| *With no egress synchronization* | | | | *With FCT protocol* | | | |
|---|---|---|---|---|---|---|---|
| *4 parallel paths* | | | | | | | |
| $BW$ [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ | $BW$ [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ |
| 100 | -14 | -63 | 17 | 100 | -2 | -12 | 11 |
| 200 | -41 | -194 | 86 | 200 | -5 | -85 | 74 |
| 300 | -35 | -6632 | 6546 | 300 | -7 | -491 | 298 |
| 400 | -13 | -141793 | 141701 | 400 | -7 | -4897 | 4526 |
| 500 | -15 | -499869 | 499478 | 500 | -7 | -13894 | 13214 |
| 600 | -106 | -543649 | 535408 | 600 | -7 | -94937 | 91991 |
| 700 | -119 | -578858 | 542561 | 700 | -7 | -254382 | 243885 |
| 800 | -38 | -683344 | 547351 | 800 | -7 | -375694 | 359466 |
| 900 | -31 | -976225 | 552171 | 900 | -7 | -466649 | 418082 |
| 1000 | -30 | -1022286 | 553458 | 1000 | -7 | -482681 | 425897 |
| *5 parallel paths* | | | | | | | |
| $BW$ [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ | $BW$ [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ |
| 100 | -2 | -17 | 13 | 100 | -2 | -18 | 17 |
| 200 | -6 | -98 | 61 | 200 | -4 | -139 | 115 |
| 300 | -21 | -3683 | 3584 | 300 | -9 | -729 | 574 |
| 400 | -11 | -159904 | 159808 | 400 | -9 | -2299 | 1575 |
| 500 | -44 | -418933 | 417555 | 500 | -9 | -18685 | 17901 |
| 600 | -22 | -434274 | 425202 | 600 | -9 | -40039 | 37833 |
| 700 | -156 | -493158 | 435529 | 700 | -9 | -121278 | 116791 |
| 800 | -42 | -852873 | 519740 | 800 | -9 | -288200 | 276433 |
| 900 | -28 | -1458732 | 736498 | 900 | -9 | -382566 | 349666 |
| 1000 | -152 | -1520055 | 762531 | 1000 | -9 | -380204 | 349372 |

Table 3: Comparison of reordering for distributed AEs with no explicit egress synchronization and with synchronization using FCT. Part 2.

and raises up to 40 $\mu$s for 6 parallel paths. This closely approaches manufacturer-stated one-way message passing latency of Myrinet configuration used for the testbed as described above.

## 4 Conclusions

When parallelizing individual AE itself, the distributed AE comprises multiple equivalent AE units running in parallel and data distribution unit, which distributes data over the multiple parallel paths in the distributed AE, and data aggregation unit, which aggregates resulting data from the parallel paths into one or more output network links.

Distributed AE brings a new problem inherent to the fact that the data is flowing through multiple independent paths—packet reordering. It results either into packet loss (if delayed out-of-order packets are just discarded) or indirectly into latency increase as the application needs to buffer the data in order sort packets before actual processing begins. For cases where better than no explicit sending synchronization is needed to minimize output packet reordering induced by the distributed AE, we have designed and evaluated Fast Circulating Token protocol providing limited synchronization among sender modules of distributed AE parallel paths. While distributed AE with no explicit sending synchronization provides limited reordering, we have shown both theoretically and experimentally that FCT decreases maximum egress packet reordering, sometimes even more than two orders of magnitude.

Regarding future work, the distributed AE could be complemented with programmable hardware implementation of load distribution and especially aggregation, which could be used for implementing sending in exact order, yielding zero packet reordering. The implementation could be based on FPGA-based programmable network interface cards with multiple interfaces.

Furthermore we would like to study different audio and video processing algorithms suitable for implementation on distributed AE. Such algorithms must minimize state sharing in order to provide efficient and scalable processing.

Finally, we would like to design an AE with built-in Quality of Service on all levels, including underlying operating system, for strict separation of different processors as well as different flows. Actually, many multicast deployment related problems, which initiated our research into overlay networking, could be mitigated if the multicast-enabled routers had strict separation of different processes running inside. In the longer term we would like to design and prototype a distributed router with such QoS capabilities.

## Acknowledgments

This project has been supported by a research intent "Optical Network of National Research and Its New Ap-

J

| *With no egress synchronization* | | | | *With FCT protocol* | | | |
|---|---|---|---|---|---|---|---|
| *6 parallel paths* | | | | | | | |
| $BW$ [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ | $BW$ [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ |
| 100 | -16 | -80 | 17 | 100 | -3 | -18 | 13 |
| 200 | -7 | -78 | 47 | 200 | -5 | -49 | 36 |
| 300 | -47 | -1955 | 1804 | 300 | -9 | -418 | 359 |
| 400 | -62 | -62744 | 62452 | 400 | -11 | -2596 | 1823 |
| 500 | -14 | -331120 | 330329 | 500 | -11 | -7553 | 6638 |
| 600 | -17 | -371489 | 357821 | 600 | -11 | -27938 | 25856 |
| 700 | -124 | -397293 | 363594 | 700 | -11 | -140596 | 135701 |
| 800 | -27 | -517597 | 379041 | 800 | -11 | -171797 | 164183 |
| 900 | -31 | -1252132 | 629255 | 900 | -11 | -324345 | 297198 |
| 1000 | -25 | -1237362 | 622667 | 1000 | -11 | -330780 | 302301 |

Table 4: Comparison of reordering for distributed AEs with no explicit egress synchronization and with synchronization using FCT. Part 3.

plications" (MŠM 6383917201) and "Parallel and Distributed Systems" (MŠM 0021622419). The authors would like to acknowledge help of Jiří Denemark and Tomáš Rebok for their assistance during measurements implementation.

## References

[1] Petr Holub, Eva Hladká, and Luděk Matyska. Scalability and robustness of virtual multicast for synchronous multimedia distribution. In *Networking - ICN 2005: 4th International Conference on Networking, Reunion Island, France, April 17-21, 2005, Proceedings, Part II*, volume 3421/2005 of *Lecture Notes in Computer Science*, pages 876–883, La Réunion, France, April 2005. Springer-Verlag Heidelberg.

[2] Petr Holub, Eva Hladká, Jiří Denemark, and Tomáš Rebok. Active elements for high-definition video distribution. In *ICT 2006, 13th International Conference on Telecommunications*.

[3] Philippe Galvez. From VRVS to EVO (Enabling Virtual Organizations). In *TERENA Networking Conference 2006*, Catania, Italy, May 2006.

[4] Markus Buchhorn. Designing a multi-channel-video campus delivery and archive service. In *The 7th Annual SURA/ViDe Conference*, Atlanta, GA, USA, March 2005.

[5] Tzi-cker Chiueh and Prashant Pradhan. Suez: A cluster-based scalable real-time packet router. In *The 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, Taipei, Taiwan, April 2000.

[6] Tzi-cker Chiueh and Prashant Pradhan. Suez: A cluster-based scalable real-time packet router. Technical Report TR-65, Experimental Computer Systems Lab, State University of New York, April 2000.

[7] D. Decasper, G. Parulkar, and B. Plattner. A scalable, high performance active network node. *IEEE Network*, 33(1):8–19, January 1999.

[8] P. Graham. A DSM cluster architecture supporting aggressive computation in active networks. In *Intl. Workshop on Distributed Shared Memory*, 2001.

[9] Young Bae Jang and Jung Wan Cho. A cluster-based router architecture for massive and various computations in active networks. In *ICOIN*, KAIST, Korea, February 2003.

[10] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel. Scalable content-aware request distribution in cluster-based network servers. In *Proc. USENIX Ann. Technical Conf.*, June 2000.

[11] R. Bianchini and E. V. Carrera. Analytical and experimental evaluation of cluster-based www servers. *World Wide Web J.*, 3(4):215–229, December 2000.

[12] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum. Locality-aware request distribution in cluster-based network servers. In *Proc. Eighth ACM Conf. Architectural Support for Programming Languages and Operating Systems*, pages 205–216, October 1998.

[13] Enrique V. Carrera and Ricardo Bianchini. Press: A clustered server based on user-level communication. *IEEE Transactions on Parallel and Distributed Systems*, 16(5):385–395, May 2005.

[14] Petr Holub. *Network and Grid Support for Mulitmedia Distribution and Processing.* PhD thesis, Faculty of Informatics, Masaryk University Brno, Czech Republic, 2005.

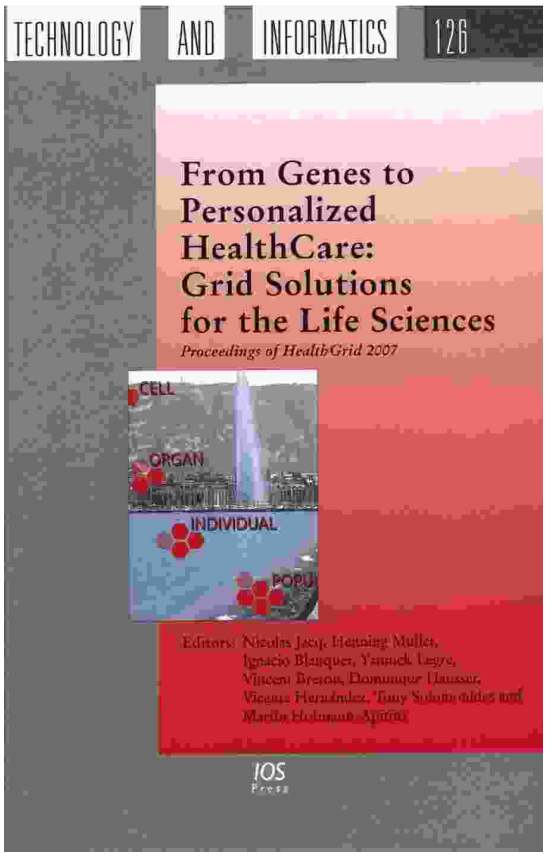
[15] Eva Hladká. *User Empovered Collaborative Environment: Active Network Support.* PhD thesis, Faculty of Informatics, Masaryk University Brno, Czech Republic, 2004.

# Appendix K

# Secure and Pervasive Collaborative Platform for Medical Applications

by Petr Holub, Eva Hladká, Michal Procházka and Miloš Liška

From Genes to Personalized HealthCare: Grid solutions for the Life Sciences, HealthGrid 2007, Geneva, Switzerland, April 2007. Proceedings. Health Technology and Informatics, Amsterdam, The Netherlands: IOS Press, 126, pp. 229–238, 10 p. ISSN 0926-9630. 2007.



# Secure and Pervasive Collaborative Platform for Medical Applications

Petr Holub[1,3], Eva Hladká[2,3], Michal Procházka[1,3], and Miloš Liška[2,3]

[1] Institute of Computer Science
[2] Faculty of Informatics,
Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
[3] CESNET z.s.p.o., Zikova 4, 160 00 Prague, Czech Republic
hopet@ics.muni.cz, eva@fi.muni.cz, michalp@ics.muni.cz, xliska@fi.muni.cz

**Abstract.** Providing secure, extensible, pervasive and easy to implement collaborative environment for medical applications poses significant challenge for state-of-the-art computer systems and networks. In this paper, we describe such a collaborative environment developed for Ithanet project, based on Grid authentication mechanisms. Significant effort has been put into developing a system, that is capable of deployment across tightly secured networking environments as implemented in vast majority of hospitals. The environment is extensible enough to incorporate Grid-service based collaborative systems like AccessGrid.

## 1 Introduction

Virtual communication environments in the medicine are slowly becoming more popular with advent of reliable high-speed networking. Current trends go in two basic directions: (1) conservative commercially available technologies like voice over IP teleconferencing and H.323/SIP infrastructures, and (2) more advanced and more experimental tools like AccessGrid. While the former group is sufficient for basic communication, we focus in this paper on more advanced extensible environments, that is suitable for advanced collaboration.

The medical network are being protected heavily namely due to requirements on security of data about patients. However, such an "adverse" networking environment has usually sever consequences on collaborative technologies. Firewalls and network address translators (NATs), together with very conservative behavior of network administrators are the major obstacles for deployment of the majority of the conferencing systems, with notable exception of Skype [1], as discussed in Section 2. For collaborative environment, synchronous transmissions audio and video signals is essential. In terms of network transfers, this is usually implemented using RTP/UDP packet distribution. But UDP packet distribution in networks with NATs and firewalls with common settings is not allowed. One way to get around this problem is to change the settings; for this, users have not only to ask for an approval of their administrators, but also they need them to make the configuration changes in security-sensitive devices, which is usually very hard to achieve. The second way is to use software for penetrating NATs and firewalls on application layer allowing communication without setting changes. This problem is not limited to collaborative environments, as demonstrated by a recent study by Open Grid Forum [2].

In this paper, we focus on building a secure extensible collaborative platform suitable for medical purposes, that is capable of working in such "adverse" networking environments inside hospitals with minimum requirements on support from network and system administrators. The platform uses Grid-compatible authentication based on PKI infrastructure and is ready to incorporate other Grid extensions based on Grid service oriented architecture.

The described system has been designed in Ithanet[4] project as stated in Acknowledgments at the end of this paper. The design reflects our experiences from participation in Ithanet and also EuroCareCF[5] projects, that showed how difficult it is to deploy reliable, secure, and affordable

[4] http://www.ithanet.eu/
[5] http://www.eurocarecf.eu/

collaborative tools in multi-institutional environment across many hospitals all over the Europe. Both according to surveys we have conducted in these projects and according to our experiences with other medical communities, there is a number of problem to solve: the connectivity among the partners varies to large extent; already existing computer systems are largely based on Microsoft Windows system (often under central administration), making installation of additional software and especially remote support very difficult or virtually impossible; computer support teams at the hospitals often do not have enough experiences to work with more complex collaborative environments. Therefore we have decided to propose and implement a system, that is able to avoid or at least to mitigate these problems.

The rest of the paper is structured as follows: Section 2 summarizes relevant related work, Section 3 gives an overview of the architecture of the whole collaborative environment, Section 5 details proposed client devices and software tools. In Section 6, we give performance measurements of the VPN-based networking solution implemented for the environment and also further experiences with the system. Section 7 concludes the paper with final remarks and directions for future development of the collaborative platform.

## 2 Related Work

Traditionally, the tele-medicine was implemented using phone-based solutions, later migrating to H.323/SIP solutions. However, both H.323 and SIP are hard to implement in "adverse" networking environments and require serious support for network administrators side.

Skype [1] has thus become very attractive alternative for many users, because of its capabilities to penetrate firewalls and work behind NATs—unless explicitly forbidden by the network administrators, and also rather robust commercial-grade implementation. Providing limited multipoint conferencing support, it is also usable for small teams to communicate. When user signs up for commercial Skype services, it is possible to call also to public telephony network. In recent versions, Skype provides basic video support, which is however much less mature than its audio counterpart. However, there are several reasons why this tool doesn't meet the needs of medical communication particularly well and becomes often explicitly forbidden by the hospital management: first, the security model is proprietary and very obscure [3, 4] and may change without any prior notification, and second, users participating in the Skype network are automatically providing their computers to be used by other Skype users, thus actually supporting the business of Skype company with their own resources. Further, it is hard to block Skype on the networking layer, because of its firewall/NAT penetration techniques (e.g., even the login messages to Skype servers may be router through the super-nodes in the P2P overlay network as shown in [4]). Being a proprietary solution, Skype is hard to be integrated in more complex Grid-like infrastructures, too.

On the other side of the collaboration tools spectra, there is an experimental open extensible system called AccessGrid [5, 6]. Since its version 2, it features service oriented architecture for the collaborative environment based on Grid services [7]. For deployment in the hospital environments, it has two major drawbacks though. First, the data distribution is not suitable for "adverse" networking environments—authors of AccessGrid assumed multicast to be the primary data distribution technology; since the multicast deployment is a problem even in very open academic networks, they added simple unicast bridging technology based on UDP packet reflectors later on. Second, the whole system is rather complicated and hard to deploy without knowledgeable on-site support staff. Despite these issues, AccessGrid features many interesting properties making in worth considering when planning deployment of larger collaborative systems, by at least preparing compatible equipment for its future installation.

Another well known example of collaboration environment, created for high energy physics community that is now becoming more widely used, is the Virtual Room Videoconferencing System (VRVS)[6]. VRVS is based on multicast-like schema and is provided as a service and user data traffic is managed by VRVS administrators. The successor of VRVS called Enabling Virtual Organizations

[6] `http://www.vrvs.org/`

(EVO)[7]—is based on self-organization of system of reflectors, again not empowering the end-user with tools to change the distribution topology. However, for medical problem it has the similar problem as AccessGrid and even worse it is a closed system with the similar consequences like for Skype.

## 3 Architecture of the Collaborative Stack

When designing the collaborative platform architecture, we have had the following main design principles in mind:

1. The system has to be secure to protect the communicated data. It should utilize Grid-compatible authentication, so that the users can use their existing Grid identity to join the collaborative environment.
2. Support for firewall and NAT penetration to facilitate deployment in "adverse" networking environments in hospitals. The support has to be flexible and auditable, and the network administrators must not be "cheated", (but rather just avoided unless really necessary).
3. The system has be extensible towards service-oriented architecture compatible with Grids.

The proposed collaborative environment comprises the following layers from the bottom-up perspective: network connectivity layer, data distribution layer, central services layer, client tools and devices layer. Each of the layers is discussed in more detail below.

*Network connectivity layer.* The network connectivity layer takes care of interconnecting all the elements of the collaborative environment into a continuous network, so that all the element may reach one another or at least some central site or server site when all-to-all connectivity is not desirable for any reason. If the elements can't reach one another over the native network, this may be implemented as an overlay network, be it simple tunnels or more sophisticated VPNs. Assuming Internet protocol stack, the tunnels may run over UDP, TCP, HTTP (including emulation of HTTP and HTML encapsulation), TCP with HTTP proxy, and TCP with SOCKS proxy. For vast majority of the firewall and NAT protected networks, at least one of the mentioned solution works. VPNs are also useful when overlay network privacy is desired based on overlay link encryption.

*Data distribution layer.* This layer takes care of multi-point data distribution to the connected clients. Assuming the Internet protocol stack, this is usually implemented either using multicast, which is more efficient but much harder to deploy properly esp. in network spanning multiple administrative domains, or using UDP packet reflectors. While less efficient, the UDP packet reflectors are much less error-prone and provide also possibility of data processing even on per-user basis (something theoretically impossible in multicast).

UDP packet reflectors may also be modified to provide the network connectivity layer to the clients and between the reflectors directly, thus merging Network connectivity and Data distribution layers.

*Central services layer.* This layer comprises services provided to the client on some "server" basis—though the servers may be largely distributed and not limited to one physically central location. The services may include monitoring, virtual rooms or venues (for creating separate virtual spaces for communication of different user groups), wikis, persistent data storage, etc.

*Client tools layer.* The client layer comprises of tools and hardware devices on client side. The software tools primarily incorporate audio, video, and a chat service, and may include other tools like shared presentation, shared desktop or application window, or shared text editor.

While software tools are traditional when looking on computer based collaborative environments, we have included also the hardware part, as the quality of hardware and a level of its

[7] `http://evo.caltech.edu/`

software support is critical for the successful experience with collaborative environments. Often this creates a point of failure for the collaborative environments deployment. The collaborative tools are much harder to deploy compared, e.g., to SETI@Home or similar computation-heavy tasks that are relatively easily distributed because they are only dependent on CPU and very basic OS services.

## 4 Preliminary Implementation of the Collaborative Stack

### 4.1 Network Connectivity Layer

Secure communication together with ability of firewall and NAT penetration is achieved using the OpenVPN software. It makes the VPN on the application layer from the ISO/OSI perspective and supports the whole range of methods as discussed in the previous section. It means that there is no need to modify configuration of network elements on the path from the client to the VPN server. Also whole communcation between client can be encrypted.

All client workstations are connected to a VPN server in a point-to-point mode. The set up of the VPN network guarantees that only the traffic belonging to the collaboration services is sent into the VPN tunnel. Originally, we wanted to use one of the private IP address ranges as defined by RFC 1918 for internal VPN addressing. After the partner networking survey we have found it very complicated to avoid conflicts with internal address ranges used at various institutions, especially as new institutions may join. The whole overlay network is therefore addressed using a public IP address range assigned by RIPE, but the addresses are treated as internal address and not distributed outside of the VPN overlay network. As there is no direct traffic between any two partners and thus all the traffic may be filtered on the VPN server.

The OpenVPN server runs in two modes—either over UDP or TCP. The UDP mode is preferred due to better performance, as the VPN is not limited by the TCP congestion control algorithm [8]. The TCP mode can also run over HTTP or SOCKS proxy.

The client side needs OpenVPN software to be able to connect to the OpenVPN server. This software makes the virtual network adapter and sets appropriate routing table records for the client. Clients are authenticated using their personal X.509 certificates—the Grid users may user their existing ones, while others are given new certificates from a dedicated certificate authority. Client software is able to work with certificates stored in the file or on the secure smart card. Second option is strongly preferred because the client certificate can be delivered and kept by the client in a secure way.

### 4.2 Data distribution layer

The data distribution layer is implemented using modular user-empowered UDP packet reflector [9], which is known to work very well with the taget client media tools—MBone Tools[8]. It is highly configurable with modules loadable in run-time, supporting sophisticated access control policing and even data transcoding for some data formats. Media streams may be encrypted by the client software tools using symmetric encryption in case that the data replication site is not considered trusted enough. Communication based on UDP packet reflector is communication with central replication unit and number of communicated client is limited by capacity of this unit. Solution of this problem is to decentralize reflector by network of reflectors [10].

### 4.3 Central services layer

Currently, there are UDP packet reflector administration and monitoring run as a central services. When the system is extended to full AccessGrid support, Venue Server could be an example of central service. Another centrally run service is the OpenVPN server supporting users as described

[8] `http://www-mice.cs.ucl.ac.uk/multimedia/software/`

in section 4.1. To support the user communication we furthermore provide an IRC daemon (currently IRCD-hybrid[9] run as a central services. A special IRC client daemon which run on the same machine was developed to store and provide the chat history. A Jabber instant messaging server may be provided in the same way as IRC.

## 5 Client Platform

The collaborative platform client is developed as a mixed HW/SW solution. The client is based on well defined and tested HW with preinstalled operating system and set of collaborative and especially videoconferencing tools. The choice of operating system was done with emphasis on simple modifications of the system, remote management, wide hardware support, security and last but not least user friendly environment. As a result we chose Ubuntu Linux to be the base for the client SW.

The collaborative tools depend on many dedicated hardware devices like sound cards, sound acquisition devices, video capture cards, USB cameras, etc., whose quality and level of support may vary to a large extent. Therefore, we have opted for suggesting a set of hardware: widely available low-cost HP Compaq dc7700 Ultra-Slim Desktop PC with known-to-work and well tested sound and graphics card and video capture cards, together with some other devices like headsets and USB cameras. This provides well defined starting point for efficient distribution of operating system and collaborative tools.

Security and personal configuration of the platform is based on combined USB security and storage token. The token is used for following purposes:

- identity storage which is based on PKI for authentication purposes, especially to connect to VPN server
- customized configuration storage for each users configuration, allowing user to store his contact informations for videoconferencing purposes and start the videoconference in specific mode (e.g., specific reflector address and port or VPN configuration)

Basic videoconferencing capabilities are provided by MBone Tools. Robust Audio Tool (RAT) is used for audio transmission and playback. RAT supports variety of audio codecs and allows to fine tune the audio stream according to quality or bandwidth limitations where necessary. Video communication is provided by Videoconferencing Tool (VIC) providing transmissions of video acquired from video capture card or USB cameras.

Besides VIC and RAT other we provide audio and videoconferencing tools like Ekiga, WengoPhone and even Skype which allow audio and video communication with number of other videoconferencing platforms.

While the individual tools are rather user-friendly when started, the initialization of the conference itself is not very intuitive step. In order to facilitate this, we are developing an integrating Graphical User Interface (GUI) (see fig. 1 for the platform, that supports easy setup of the conference. The default settings may be stored for future reuse, so in the production state, the user just pushes a single button to start the whole conference. The GUI also monitors all the applications, so that if any part of the system crashes, the user is immediately informed and it also provides a very detailed information for the remote user support.

To support the collaboration beyond scope of just audio or videoconferencing we provide bidirectional sharing of whole desktop or particular applications between videoconferencing platform clients. Desktop and application sharing is based on VNC protocol [11] and related tools, namely shared-app-vnc[10] and x11vnc[11]. A secondary intent is to provide user with remote control of his videoconferencing machine in case the machine has no display and keyboard or the user wants to

[9] `http://ircd-hybrid.com/`
[10] `http://shared-app-vnc.sourceforge.net/`
[11] `http://www.karlrunge.com/x11vnc/`

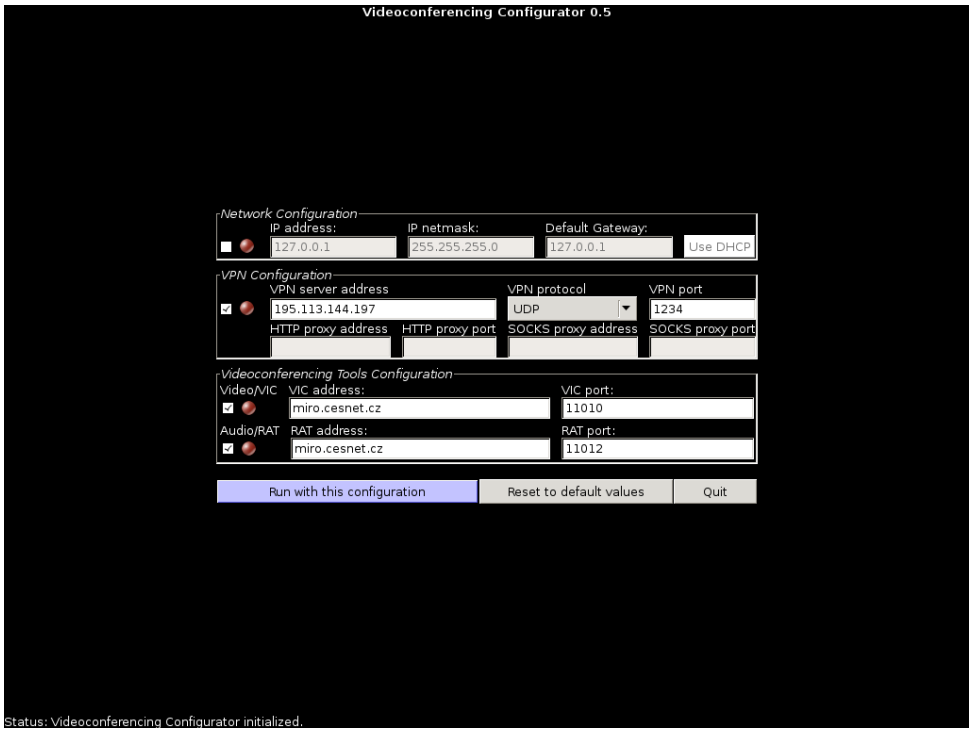



**Fig. 1.** GUI for the collaborative platform client.

control the videoconference from his/her laptop computer. However, using VNC as a software display is considered as an emergency solution only because of high demands on the videoconferencing machine and the latency introduced video displaying.

The client platform is based on SW that is stable yet under development and some new and desired functionality may appear. In a worse case a security hole may be discovered in one or more platform SW components. Thus it is necessary to update the SW base of the platform regularly. The individual videoconferencing tools as well as other platform SW may be updated automatically based on the operating system SW repositories.

Underlying operating system updates are more complicated and failed update may turn whole machine unusable. There may be no system administrator available or skilled enough to perform the operating system update on end users site. That is why we opt for prepare the operating system update as a black box solution. The update is based on bootable CD/DVD with an image of well tested and working updated operating system and platform SW. It is not necessary to distribute the CDs among the end users. More comfortable approach is to create and ISO image of the update CD and make it available for download. The update procedure is performed automatically right after the CD is inserted into the machine. End users don't need to reconfigure their box after each operating system update because all custom configuration is stored on the USB token and thus is not affected by the update.

## 6 Experiences with Preliminary Implementation

In order to evaluate influence of incorporation of OpenVPN into the collaborative platform, we have measured a number of parameters critical for real-time multimedia communication using different VPN modes. The measurement testbed comprised one client and one VPN server, interconnected with high-speed backbone network link with capacity above 1 Gbps spanning about 250 km. The

results of measurements are summarized in Table 1. We can conclude that UDP based VPN is very safe and has minimum impact on the traffic. Slight CPU requirements increase for the UDP based VPN compared to TCP based VPN is due to application-level packet loss recovery and congestion control, which is marginally less CPU efficient compared to kernel-based TCP implementation. TCP-based VPNs also perform very well provided they are on low-latency network with very low packet loss, so that congestion control algorithm doesn't influence the data flow significantly. If the HTTP proxy is of good performance, it has minimum impact on performance, too.

**Table 1.** Measured comparison between direct communication and communication through various VPN modes as implemented by OpenVPN.

| | no VPN | UDP VPN | TCP VPN | TCP VPN + HTTP proxy |
|---|---|---|---|---|
| pchar latency [ms] | 3.51 | 3.69 | 3.94 | 3.93 |
| iperf jitter [$\mu$s] | 6 | 6 | 9 | 13 |
| pchar capacity est. [Mb/s] | 39.8 | 35.2 | 20.1 | 19.8 |
| iperf packet loss @ 30 Mb/s [%] | 0.0 | 0.0 | 0.0 | 0.0 |
| iperf CPU idle @ 30 Mb/s [%] | 48.9±0.2 | 41.7±0.4 | 44.5±0.4 | 42.6±0.4 |

When evaluating the performance of this solution subjectively, the media streams are fine and the overall quality is very good. The only problem we were facing is that the users are sometimes very reluctant to buy a new hardware for the client platform, even though it is very cost effective compared to dedicated videoconferencing solutions. However, they are also unhappy about performance of videoconferencing tools self-installed on their existing desktop computers, as these were not performing well without substantial tweaking because of rather complex interactions with undefined or poor hardware components and existing software. Thus the deployment requires significant work in order to explain principles of the system to the users as described above.

K

## 7 Conclusions and Future Work

In this paper, a secure and pervasive collaborative platform for medical applications has been introduced to provide flexible multi-point Grid-compliant collaborative environment. The design and implementation was targeted to create remotely supported system, that is scalable, robust and flexible allowing to collaborate to tens of people.

The first version of the system, described in this paper, has risen a number of new problems and ideas. The first ideas for future work are concerned with reflector functionality. In the future we plan to utilize per-user processing on the reflector for solving the problem when a single client with a very limited network connectivity limits the quality of the collaboration for the whole group. Currently we are using OpenVPN to traverse NATs and firewalls but in the future, we plan to implement this directly in the reflectors. Such an approach allows for more aggressive failure detection and faster problem recovery. Also as the reflectors may be deployed as a network, it naturally avoids the single point of failure currently imposed by a central VPN server.

Grids and Grid-based systems are now widely developed and used in many areas. We plan to utilize the Grid-services based approach and to incorporate AccessGrid services. On the other side, AccessGrid needs to be modified to work with our advanced reflectors, firewall and NAT penetration techniques and reflector networks for better scalability and robustness. We are at the beginning of practical usage of the proposed platform and its routine operation will definitely bring other new ideas and requirements for the future.

## 8 Acknowledgments

This work has been kindly supported byt European Commission project "ITHANET – eInfrastructure for Thalassaemia Research Network", RI-2004-026539.

## References


1. Zennström, N., Friis, J.: Skype (2003-2007) http://www.skype.com/.
2. Niederberger, R., Allcock, W., Gommans, L., Grünter, E., Metsch, T., Monga, I., Volpato, G.L., Grimm, C.: Firewall issues overview. Technical Report GFD-I.083, Open Grid Forum (2006)
3. Biondi, P., Desclaux, F.: Silver needle in the skype. In: BlackHat Europe. (2006) http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-biondi/bh-eu-06-biondi-up.pdf.
4. Baset, S.A., Schulzrinne, H.: An analysis of the skype peer-to-peer internet telephony protocol. In: INFOCOM 2006, Barcelona, Spain (2006) http://www1.cs.columbia.edu/~salman/publications/skype1_4.pdf.
5. Childers, L., Disz, T., Hereld, M., Hudson, R., Judson, I., Olson, R., Papka, M.E., Paris, J., Stevens, R.: ActiveSpaces on the Grid: The construction of advanced visualization and interaction environments. In Engquist, B., ed.: Simulation and visualization on the grid: Parallelldatorcentrum, Kungl. Tekniska Högskolan, seventh annual conference, Stockholm, Sweden, December 1999: proceedings. Volume 13 of Lecture Notes in Computational Science and Engineering., New York, NY, USA, Springer-Verlag Inc. (2000) 64–80
6. Childers, L., Disz, T., Olson, R., Papka, M.E., Stevens, R., Udeshi, T.: Access grid: Immersive group-to-group collaborative visualization. In: Proceedings of Immersive Projection Technology, Ames, Iowa (2000)
7. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The physiology of the grid: An open grid services architecture for distributed systems integration (2002)
8. Jacobson, V.: Congestion avoidance and control. In: ACM SIGCOMM '88, Stanford, CA (1988) 314–329
9. Hladká, E., Holub, P., Denemark, J.: An active network architecture: Distributed computer or transport medium. In: 3rd International Conference on Networking (ICN'04), Gosier, Guadeloupe (2004) 338–343
10. Holub, P., Hladká, E., Matyska, L.: Scalability and robustness of virtual multicast for synchronous multimedia distribution. In: Networking - ICN 2005: 4th International Conference on Networking, Reunion Island, France, April 17-21, 2005, Proceedings, Part II. Volume 3421/2005 of Lecture Notes in Computer Science., La Réunion, France, Springer-Verlag Heidelberg (2005) 876–883
11. Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual network computing. IEEE Internet Computing **2** (1998) 33–38

# Appendix L

# Distributed Active Element in 10 Gbps Network

by Petr Holub and Eva Hladká

In 13th International Conference on Telecommunications, ICT 2007, Penang, Malaysia, May 2007. Proceedings. IEEE/MICC, 2007. pp. 1-6, 6 p. ISBN 1-4244-1094-0.

# Distributed Active Element in 10 Gbps Network

***Abstract*—In this paper, we propose a distributed Active Element for tightly coupled cluster environment, suitable for distribution of large bandwidth data that exceed capacity of every single node in the cluster. This approach utilizes the fact that real-time multimedia transmission systems relying on non-guaranteed protocols like UDP need to handle limited packet reordering on their own. We describe the Fast Circulating Token protocol, which enables imposing even stricter bound on the outbound packet reordering. The whole system is examined on 10GE testbed and shows very good performance. The FCT provides expected improvement, making the packet reordering comparable to long haul networks.**

***Index Terms*—multi-point user-empowered data distribution, distributed Active Elements, virtual multicast, multimedia data distribution**

## I. Introduction

High-speed networking proliferation has catalyzed development and deployment of new real-time communication systems based on distribution of high-bandwidth multimedia information, like 4K and High-Definition (HD) video systems [1], [2], [3]. When designed for multi-point collaborative environments, multi-point data distribution needs to be ensured. Native network multicast has turned to perform less then acceptably in many cases and thus systems like Flexcast [1] or reflectors [2] have been developed. In this paper, we focus on a parallel reflector-based system called distributed Active Element (AE) for synchronous multimedia data distribution and processing [4], [5], which allows distribution of high bandwidth streams using parallel processing on tightly coupled commodity computer clusters. Parallel processing is incorporated in specialized hardware of modern high-performance switches and routers. However, its implementation on general purpose hardware is limited because of requirement for zero output packet reordering, which has severe consequences, e.g., on slowing down performance of the most widely used transport protocol—TCP [6]. The real-time multimedia communication relying on non-guaranteed transmission protocols, like RTP over UDP, however needs to handle the packet reordering on their own anyway, as the reordering is often present in long-distance transmissions for high-bandwidth data rates [6], [7]. We examine performance of the distributed AE in 10 Gigabit Ethernet environment for distribution of multi-Gigabit data streams, as used by advanced multimedia systems relying on uncompressed HD and post-HD video.

*Related work.* Since the native multicast support is not always available, reliable, or performing well enough, multicast virtualization technologies have been introduced, like the H.323 MCUs or reflectors in the Virtual Room Videoconferencing System (VRVS)[1]. Its successor EVO [8] is based on self-organization of system of reflectors. Similar approach has been pursued earlier by our group [9]. Other simpler UDP packet reflectors include rcbridge [10], reflector[2], and Alkit Reflex[3].

Another area related to this paper is utilization of computer clusters as either distributed routers or distributed servers [11], [12], [13]. Project Suez [14] is a distributed router based on commodity PC cluster with Myrinet interconnection with each node of the cluster having one internal interface to the Myrinet switch and optionally one or more external interfaces. Suez uses a routing-table search algorithm that exploits CPU cache for fast lookup by treating IP addresses directly as virtual addresses. Another project which distributes processing load on active network elements is Active Network Node [15] that relies on specialized hardware. Software DSM project [16] attempts to build efficient distributed memory for closely coupled clusters for using them as active routers. There is yet another similar project called Cluster-based Active Network Router [17]. However none of the above mentioned projects addresses finer than per-address network load distribution—thus there is no need for solving packet reordering issues, but on the other hand it doesn't solve the problem of data distribution for streams where the bandwidth exceeds capacity of each single parallel unit in the cluster.

Our distributed AE is designed not only for data distribution, but allows for processing as well. Relevant parallel programming paradigm for stream processing on distributed clusters has been proposed in MIT StreamIt project [18].

*Paper organization.* Section II gives a brief overview of distributed AE architecture, introducing the packet reordering limiting using Fast Circulating Token protocol. Section III describes experimental results of distributed AE used with multi-Gigabit data flows. Concluding remarks and future work ideas are given in Section IV.

## II. Distributed Active Element

In this section, we give an overview of the distributed AE and its basic properties with respect to data distribution, required for understanding experimental section of this paper. More details on the distributed AE can be found in [4], [5]. The architecture of the distributed AE is based on architecture of AE [9] and it is partly determined by requirement of implementability on existing tightly coupled clusters with low latency interconnection. The computing nodes form a computer cluster with each node having two connections: (1) *low-latency control connection* used for internal communication

[1] http://www.vrvs.org/

[2] http://www.cs.ucl.ac.uk/staff/s.bhatti/teaching/z02/reflector.html

[3] http://w2.alkit.se/reflex/

and synchronization inside the distributed AE, and (2) *data connection* used for receiving and sending the data. Example of such a system is shown in the evaluation testbed description in Figure 2.

The incoming data needs to be first distributed across the multiple parallel units of the distributed AE, processed in these units, and finally aggregated and sent over the network to the listening clients. Thus the architecture comprises three major parts:

- *Distribution unit* takes care of ingress data flow distribution over multiple parallel distributed AE units. When the distribution unit is part of the same L2 domain as parallel AE units (e.g., using VRRP or CARP protocols), it may operate on L2 addresses only, otherwise L3 (usually IP) addressing is needed.
- *Parallel AE unit* is a complete instance of AE with modified sender module to allow for possible synchronization. It has the kernel with administrative submodules, session management, processor schedulers, and AAA submodules. Data is received by network listener modules, stored into shared memory (shared across a single instance of the reflector only, not across multiple AE unit instances), processed by zero or more processors, distribution lists filled up with either one of processors or with session management and finally sent with the sender module. The network management module handles communication with distribution unit and also communication with other distributed AE units.
- *Aggregation unit* aggregates the resulting traffic to output network line(s). Because the AE element is most commonly used for data multiplication, we assume that output data flow from the distributed AE is larger than input data flow. Thus we need a unit that is even more powerful than the input load distribution unit. In most cases, cheap custom made software implementation is not available and we have to use available hardware solution like aggregating switch. However, in that case we must not assume any further behavior of the aggregating unit except for the following two things: first, it is over-provisioned enough not to loose any data, and second, it has limited buffer space available.

There are also protocols designed for set up and maintenance the distributed AE [5], so that new parallel units may join in and existing may leave.

- The *ideal network* is a network in which no data is lost, corrupted, nor reordered. It also provides instant delivery, i. e., it introduces zero latency.
- The *ideal multimedia traffic* has bandwidth $b$ and independent packets of exactly same size $s_p$, which is also used to express all the queue sizes in the system. In order to isolate reordering introduced by the distributed AE, we assume that the ideal multimedia traffic has no reordering prior to entering distribution unit.
- The *ideal aggregating unit* has $n$ identical input interfaces and a single output interface with capacity equal or bigger than the $n$ inputs together. It reads packets from the size-limited input interface queues and sends them on an output interface in such a way, that packets are never lost. The speed is $b_j^{SW}$ (bits per second) for $j$-th input interface and each input queue has equal size of $s_i^{SW}$ for each input interface. In order not to lose any input data, the ideal aggregating unit needs to fulfill the following requirement in the steady state: $\sum_j b_j^{SW} \leq b_o^{SW}$.
- The *ideal AE* has processing capacity equal or higher than stream bandwidth and it has an input queue size of $s_i^{AE}$ and all the parallel units have the same parameters and performance. The ideal AE introduces no losses, no data corruption, nor data reordering in the data stream.

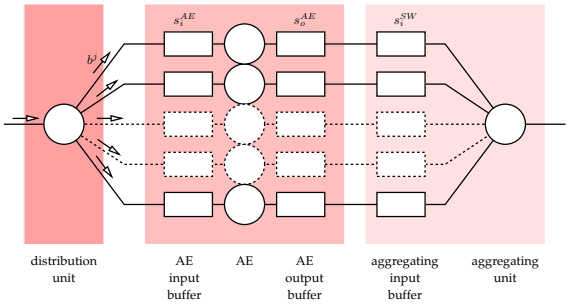



Figure 1. Model of the ideal distributed AE with ideal aggregation unit.

### A. Ingress Distribution

The ingress data distribution takes care of distributing incoming data across different paths inside the distributed AE. For the ideal distributed AE the ideal distribution unit distributes packets in round-robin fashion. In each round, it distributes $m$ packets, one to each of the parallel units. The distribution unit marks round number into each packet. This protocol is modified in case that parallel units are of unequal performance.

### B. Egress Synchronization

*1) No explicit synchronization:* The simplest model for egress synchronization is to use no synchronization at all. However, with this model and limited buffers on the input interfaces of the AEs, there is still some implicit synchronization achieved.

It can be shown the maximum reordering induced by an ideal distributed AE (shown in Figure 1) with no explicit egress synchronization and ideal aggregating unit is

$$n(s_i^{AE} + s_o^{AE} + s_i^{SW} + 1),$$

where $n$ is the number of parallel AE units when all queues operate in FIFO tail-drop mode. Detailed proof can be found in [5].

*2) Fast Circulating Token:* In order to decrease packet reordering introduced by the distributed AE, we have introduced a distributed algorithm for achieving less packet reordering compared to no explicit synchronization. The nodes are ordered in a ring with one node elected as a master node and they circulate a token which serves as a barrier so that no node can run too much ahead with sending data. After reception of the token containing the current "active" round number, each non-master node passes on the token immediately and may send only the data from the round marked in the token until it receives to token again. When the master node receives the token from the last node in the ring, it finishes sending the current round, increments the round number in the token a passes on the token. The mechanism is called *Fast Circulating Token* (FCT) since the token is not held for the entire time period of data sending as usual in the token ring networks.

Because of real world implementation of data packet sending in common operating systems, we assume that sending procedure for a single packet is non-preemptive. Further we assume that token reception event processing has precedence over any other event processing in the distributed AE. However, as the data sending is non-preemptive, if the token arrives in the middle of data packet sending, it will be handled just after that packet sending is finished.

After more detailed analysis [5], it can be shown the maximum reordering induced by an ideal distributed AE with FCT egress synchronization and ideal aggregating unit is

$$n(s_i^{SW} + 3),$$

where $n$ is the number of parallel AE units when all queues operate in FIFO tail-drop mode.

When operating in a non-ideal environment, there are several complications that need to be taken into account:

- packet reordering, either before data reaches the distributed AE, or on a single parallel path inside distributed AE—settings of the FCT determine whether excessive packet reordering will be converted to packet loss or not,
- due to unequal performance of parallel paths, load balancing may be deployed—again the reordering of two consecutive packet is limited by size of two consecutive rounds, but each round may have more than $n$ packets depending on load balancing scheme used,
- packet loss due to overloading of distributed AE or some of its parts.

*3) Exact Order Sending:* It is possible to design sending protocol that results into exact ordering, but it requires defined behavior of aggregation unit and thus it is not suitable for implementation on commodity hardware like aggregating switches. The details can be found in [4], [5].

## III. Prototype Performance Evaluation in 10GE Environment

Prototype implementation of the distributed AE is implemented in ANSI C language for portability and performance reasons. The implementation comprises two parts: a load distribution library and the distributed AE itself.

Because of lack of flexible enough load distribution hardware unit, we have implemented it as a library, which allows simple replacement of standard UDP related sending functions in existing applications and allows developers to have defined type of load distribution—either pure round robin or load balancing.

Each parallel AE uses threaded modular implementation based on architecture described in Section II. Internal buffering capacity of each AE node has been set to 500 packets. Explicit synchronization using FCT protocol has been implemented using MPICH implementation[4] of MPI built with low-latency Myrinet GM 2.0 API[5] (so called MPICH–GM). Prototype implementation has been tested on Linux.

For cost-effective prototype implementation, the aggregation unit was a implemented as commodity switch with sufficient capacity of internal switching matrix.

### A. Experimental Setup

The behavior of the distributed AE has been evaluated on a 10GE testbed shown in Figure 2. The sender and receiver machines were identical PCs with dual AMD Opteron 250 processor at 2.4 GHz, 2 GB RAM, and 10GE Chelsio T110 NIC card in a PCI-X 133 MHz slot. Both computers were running SuSE Linux 9.1 with 2.6.6 vanilla kernel with Chelsio drivers and patches. The computers were connected to 10GE ports of the Cisco 6506 switch.

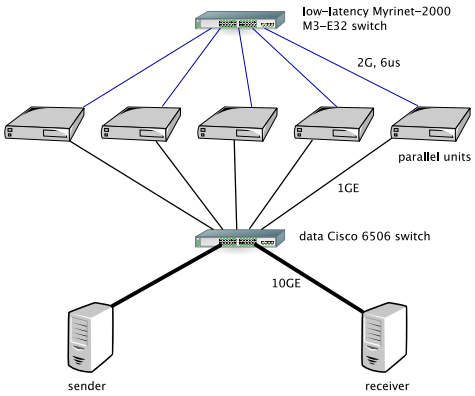



Figure 2. Experimental testbed setup.

The parallel AE units were run on a IA32 PC cluster with Myrinet-2000 low-latency interconnection. The nodes were equipped with dual Intel Xeon at 2.4 GHz, 2 GB RAM, and Broadcom NetXtreme BCM5703 card plugged into Gigabit Ethernet ports of Cisco 6506 switch. Each node had also Myrinet M3F-PCI64C-2 NIC plugged into Myrinet M3-E32 with M3-SW16-8F interface.

[4]http://www-unix.mcs.anl.gov/mpi/mpich/
[5]http://www.myri.com/scs/GM-2/doc/html/

| *# parallel units* | *max. bw [Mbps]* |
|---|---|
| 1 | 800 |
| 2 | 1600 |
| 3 | 2400 |
| 4 | 3200 |
| 5 | 4000 |
| 6 | 5000 |
| 7 | 5000 |
| 8 | 5000 |

Table I
MAXIMUM FORWARDING BANDWIDTH FOR VARYING NUMBER OF PARALLEL UNITS FOR THE DISTRIBUTED AE.

The distribution unit was implemented in software based on direct IP addressing of the parallel nodes, the stream splitting was carried out by the 6506 switch, which was also acting as the aggregating unit.

### *B. Performance Evaluation*

In order to evaluate raw performance of the AE, we have measured maximum bandwidth of a stream that the AE is able to forward without packet loss greater than 0.1%. This benchmark more demanding and thus also representative because of memory to memory copying limitations, compared to the replication, which easily saturates the bandwidth of the outgoing network interface [19].

*a) Stand-alone reflector:* Standalone reflector running on one node of the testbed cluster was measured for reference purposes and it was able to forward data up to 800 Mbps.

*b) Distributed AE:* Maximum bandwidth for forwarding with less than 0.1% packet loss was measured for varying number of parallel units and the results are summarized in Table I. Note the saturation at 6 parallel units, which is caused by the maximum throughput on the Chelsio T110 cards on sender and receiver.

For low loss area, the results are equivalent both for the unsynchronized and FCT-synchronized version of the distributed AE. It also turns out that the performance scales linearly with respect to the number of parallel AE units. When examining the higher loss areas (which are not usable for real data distribution anyway), the FCT-synchronized version performs slightly worse than the unsynchronized. This can be observed from upper part of the graphs in Figure 3, for 2 and 4 parallel units respectively; the lines are overlapping for 6 and 8 units as the higher loss region is not reached because of sender/receiver saturation as discussed above.

### *C. Packet Loss and Reordering Evaluation*

The reordering is expressed as the difference between sequence numbers of two consecutive packets. Thus, if all the sequentially numbered packets arrive in the same order they were sent, all the differences are +1. Higher number than +1 means, that some packets were skipped forth (either because of packet reordering or because of packet loss) while negative number means stepping back in packet numbering (due to packet reordering only). Value of 0 occurs when duplicate packets arrive immediately following each other. $\min\{j\}$ is the maximum negative difference in sequence numbers of successively received packets. The $\min\{j\}$ is very important from the application developer perspective, as it gives the amount of packet buffer needed to reconstruct the proper order of packets (provided no packet loss occurs), and also from the users perspective, as the amount of buffering it related to latency increase the users are experiencing.

| *BW* | FCT-sync | | | unsync | | |
|---|---|---|---|---|---|---|
| [Mbps] | $\min\{j\}$ | $H^-$ | $N^-$ | $\min\{j\}$ | $H^-$ | $N^-$ |
| 200 | 0 | 0 | 0 | -4 | -12 | 3 |
| 400 | 0 | 0 | 0 | -6 | -7 | 2 |
| 600 | -1 | -6 | 6 | -3 | -5 | 2 |
| 800 | -5 | -26 | 16 | -4 | -13 | 5 |
| 1000 | -6 | -124 | 93 | -10 | -21 | 5 |
| 1200 | -7 | -82 | 67 | -33 | -113 | 15 |
| 1400 | -5 | -220 | 179 | -39 | -455 | 65 |
| 1600 | -6 | -522 | 466 | -7 | -301 | 281 |
| 1800 | -5 | -1162 | 1104 | -50 | -911 | 627 |
| 2000 | -5 | -1317 | 1252 | -20 | -1214 | 1119 |
| 2200 | -7 | -1545 | 1443 | -10 | -1706 | 1599 |
| 2400 | -7 | -2634 | 2520 | -14 | -2591 | 2435 |
| 2600 | -6 | -4946 | 4736 | -14 | -5423 | 5177 |
| 2800 | -15 | -6539 | 5886 | -15 | -7351 | 7005 |
| 3000 | -6 | -7963 | 7424 | -31 | -10987 | 9654 |
| 3200 | -7 | -8712 | 7117 | -89 | -9420 | 7592 |
| 3400 | -7 | -9431 | 5060 | -100 | -9104 | 4827 |
| 3600 | -7 | -57523 | 27730 | -38 | -50178 | 26252 |
| 3800 | -7 | -256152 | 122298 | -111 | -253093 | 125121 |
| 4000 | -7 | -482062 | 229886 | -23 | -480988 | 236434 |
| 4200 | -7 | -989134 | 464849 | -33 | -952629 | 476268 |
| 4400 | -7 | -1484894 | 685827 | -46 | -1535218 | 755956 |
| 4600 | -7 | -1081210 | 497005 | -25 | -938601 | 473483 |
| 4800 | -7 | -406902 | 183817 | -43 | -181140 | 97237 |
| 5000 | -7 | -38077 | 19546 | -27 | -13435 | 10480 |

Table II
PACKET REORDERING FOR 8 PARALLEL UNITS.

For any interval of arrivals of two or more packets, the following equation holds

$$\underbrace{\sum_{j=\min\{j\}}^{-1} jh_j}_{H^-} + \underbrace{h_1}_{H^1} + \underbrace{\sum_{j=2}^{\max\{j\}} jh_j}_{H^+} = \Delta, \qquad (1)$$

where $\Delta$ is difference between sequence number of last and first packet in the observed interval. Also, for the any interval of arrivals of more than one packet, the following equation holds:

$$\Pi + \underbrace{\sum_{j=\min\{j\}}^{-1} h_j}_{N^-} + \underbrace{h_1}_{N^1} + \underbrace{\sum_{j=2}^{\max\{j\}} h_j}_{N^+} - \delta = \Delta. \qquad (2)$$

where $\Pi$ is number of lost packets and $\delta$ is a number of duplicated packets that are not included in $h_0$. Proofs for both can be found in [5]. By combining both equations (1) and (2), we can derive packet loss as $\Pi = H^- + H^+ - N^+ - N^- + \delta$. Because positive part of the graph described by $H^+$ or $N^+$ includes also packet loss, the negative part of the graph described by $H^-$ or $N^-$ can be seen as measure of packet reordering.

The difference between the $H$-sums and the $N$-sums is that the $H$-sums are "weighted sums". Thus the more packets are farther from 1 in either direction, the higher the absolute value of $H$-sums are, while the $N$-sums remain the same. All the terms in the $N^-$, $N^+$, and $H^+$ are positive and all the terms in the $H^-$ are negative. If $H^- \approx N^-$, the vast majority of out-of-order packets in the negative part is reordered by $j = -1$.

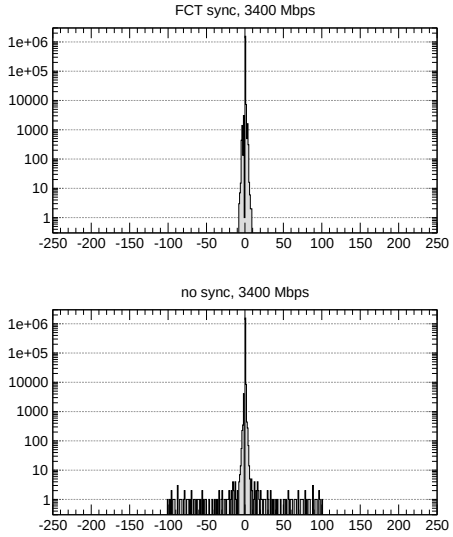



Figure 4. Sample packet reordering distribution with FCT and without synchronization, for 8 parallel units and 3.4 Gbps per data flow.

*c) Stand-alone reflector:* The bigger the loss is above the 800 Mbps performance limit, the larger the sum $H^+$ is. No reordering nor duplicates are introduced, thus $H^- = N^- = h_0 = 0$.

*d) Distributed AE:* The results presented here are a subset of complete set of measurement carried out in order to evaluate the behavior of distributed AE throughly. Figure 4 shows dependence of $\min\{j\}$ on bandwidth of the forwarded stream (lower part of each graph) together with packet loss (upper part) for 2, 4, 6, and 8 parallel units. The behavior for 3, 5, and 7 is comparable. It turns out that before the saturation of the distributed AE (c.f. Table I), the FCT significantly improves maximum packet reordering $\min\{j\}$.

More detailed results for 8 parallel units is shown in Table II, revealing that $H^-$ and $N^-$ are similar for FCT and unsynchronized versions. Sample reordering distribution for the 3400 Mbps stream and 8 parallel units is given in Figure 4. The reordering results are appropriate to be viewed in the context of long haul network paths. Given examples [7] of rather problematic Washington D. C. to Los Angeles link with $\min\{j\}$ over -60 and high-quality link from Los Angeles to Pittsburgh with $\min\{j\}$ of -1, the FCT gives much better results than the former link and gives comparable results to the latter one. Thus a real-time multimedia transmission application, which has been developed to work over long distance networks, needs to adapt to even significantly worse packet reordering than the one outgoing from distributed AE, to perform reliably in real world conditions.

Token round-trip time in FCT protocol has been also monitored and it ranges between 14 $\mu$s for 2 parallel AE paths and raises up to approximately 60 $\mu$s for 8 parallel paths, which is in accordance with one-way message passing latency of Myrinet configuration used for the testbed as described above.

## IV. Conclusions

In this paper, we have presented the concept of distributed Active Element, which relaxes the requirement for strict packet ordering, assuming that the real-time multimedia transmission applications relying on non-guaranteed protocols like UDP need to adapt to some degree of packet reordering on their own. This allowed us to design and prototype a scalable system for data distribution and potentially also processing of real-time multimedia data based on tightly coupled clusters with low-latency internal interconnection. We have proposed the Fast Circulating Token protocol in order to impose harder upper bound on the maximum packet reordering. The prototype system has been examined using 10 Gigabit Ethernet testbed and the results suggest its usability for high-end applications.

In the future, we would like to focus on three basic areas. First, we would like to adapt some of the parallel stream processing paradigms like StreamIt [18] to program data processing for the distributed AE, thus turning the system into a more general active router. Second, having the data processing, we would like to extend the distributed AE with quality of service support on several levels (network bandwidth, CPU capacity, memory capacity and bandwidth, etc.). Third, we intend to examine suitability of custom hardware solutions based on FPGA to build a aggregation unit allowing exact order packet sending.

## Acknowledgments

This project has been supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201) and "Parallel and Distributed Systems" (MŠM 0021622419). The authors would like to acknowledge help of Jiří Denemark and Tomáš Rebok for their assistance during measurements implementation.

## References

[1] T. Shimizu, D. Shirai, H. Takahashi, T. Murooka, K. Obana, Y. Tonomura, T. Inoue, T. Yamaguchi, T. Fujii, N. Ohta, S. Ono, T. Aoyama, L. Herr, N. van Osdol, X. Wang, M. D. Brown, T. A. DeFanti, R. Feld, J. Balser, S. Morris, T. Henthorn, G. Dawe, P. Otto, and L. Smarr, "International real-time streaming of 4K digital cinema," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 929–939, Oct. 2006.

[2] P. Holub, L. Matyska, M. Liška, L. Hejtmánek, J. Denemark, T. Rebok, A. Hutanu, R. Paruchuri, J. Radil, and E. Hladká, "High-definition multimedia for multiparty low-latency interactive communication," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 856–861, 2006.

[3] J. Jo, W. Hong, S. Lee, D. Kim, J. Kim, and O. Byeon, "Interactive 3D HD video transport for e-science collaboration over UCLP-enabled GLORIAD lightpath," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 884–891, 2006.

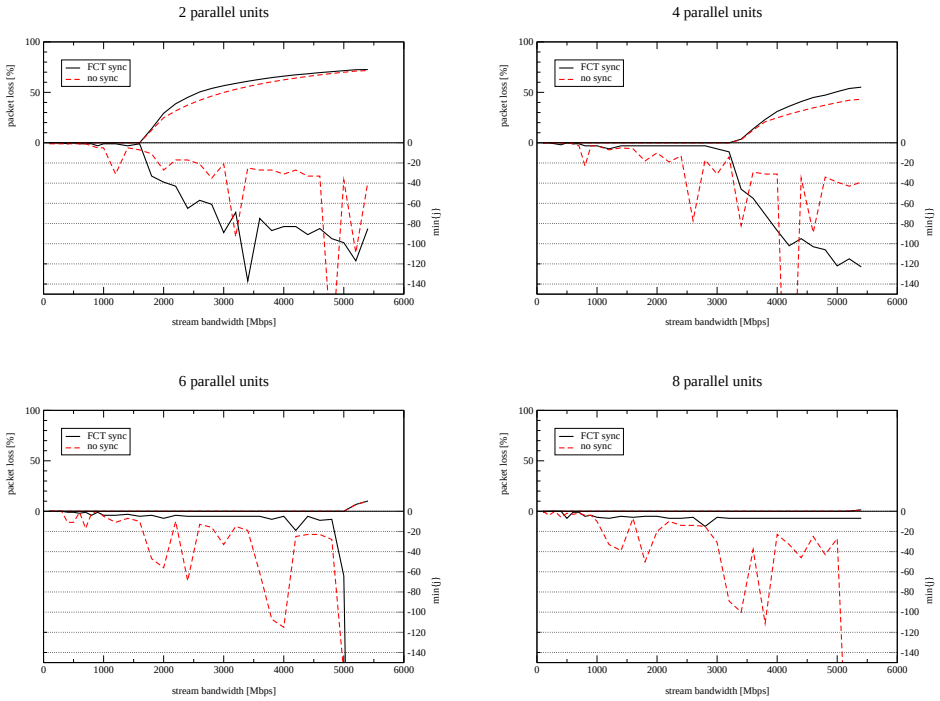



Figure 3. Packet loss and reordering comparison for FCT synchronized and unsynchronized version for distributed AE and various number of parallel units.


[4] P. Holub and E. Hladká, "Distributed active element for high-performance data distribution," in *NPC 2006: Network and Parallel Computing*, Tokio, Japan, Oct. 2006, pp. 27–36.

[5] P. Holub, "Network and grid support for mulitmedia distribution and processing," Ph.D. dissertation, Faculty of Informatics, Masaryk University Brno, Czech Republic, 2005.

[6] J. C. R. Bennett, C. Partridge, and N. Shectman, "Packet reordering is not pathological network behavior," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 789–798, Dec. 1999.

[7] L. Gharai, C. Perkins, and T. Lehman, "Packet reordering, high speed networks and transport protocol performance," in *Proceedings of the 13th International Conference on Computer Communications and Networks (ICCCN'04)*, Chicago, IL, USA, Oct. 2004, http://ultragrid.east.isi.edu/publications/2004icccn.pdf.

[8] P. Galvez, "From VRVS to EVO (Enabling Virtual Organizations)," in *TERENA Networking Conference 2006*, Catania, Italy, May 2006.

[9] P. Holub, E. Hladká, and L. Matyska, "Scalability and robustness of virtual multicast for synchronous multimedia distribution," in *Networking - ICN 2005: 4th International Conference on Networking, Reunion Island, France, April 17-21, 2005, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 3421/2005. La Réunion, France: Springer-Verlag Heidelberg, Apr. 2005, pp. 876–883. [Online]. Available: http://www.springerlink.com/index/GETV62MG4GA0CUPL

[10] M. Buchhorn, "Designing a multi-channel-video campus delivery and archive service," in *The 7th Annual SURA/ViDe Conference*, Atlanta, GA, USA, Mar. 2005.

[11] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum, "Locality-aware request distribution in cluster-based network servers," in *Proc. Eighth ACM Conf. Architectural Support for Programming Languages and Operating Systems*, Oct. 1998, pp. 205–216.

[12] R. Bianchini and E. V. Carrera, "Analytical and experimental evaluation of cluster-based WWW servers," *World Wide Web J.*, vol. 3, no. 4, pp. 215–229, Dec. 2000.

[13] E. V. Carrera and R. Bianchini, "Press: A clustered server based on user-level communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 5, pp. 385–395, May 2005.

[14] T. Chiueh and P. Pradhan, "Suez: A cluster-based scalable real-time packet router," in *The 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, Taipei, Taiwan, Apr. 2000.

[15] D. Decasper, G. Parulkar, and B. Plattner, "A scalable, high performance active network node," *IEEE Network*, vol. 33, no. 1, pp. 8–19, Jan. 1999.

[16] P. Graham, "A DSM cluster architecture supporting aggressive computation in active networks," in *Intl. Workshop on Distributed Shared Memory*, 2001. [Online]. Available: http://www.cs.umanitoba.ca/~pgraham/papers/DSM_body.pdf

[17] Y. B. Jang and J. W. Cho, "A cluster-based router architecture for massive and various computations in active networks," in *ICOIN*, KAIST, Korea, Feb. 2003. [Online]. Available: http://camars.kaist.ac.kr/~ybjang/research/publication/icoin2003.pdf

[18] W. Thies, M. Karczmarek, and S. Amarasinghe, "Streamit: A language for streaming applications," in *Proceedings of the 11th International Conference on Compiler Construction*, ser. Lecture Notes in Computer Science, vol. 2304/2002. Grenoble, France: Springer-Verlag Heidelberg, 2002, pp. 179–196.

[19] E. Hladká, "User empovered collaborative environment: Active network support," Ph.D. dissertation, Faculty of Informatics, Masaryk University Brno, Czech Republic, 2004.

# Appendix M

# Virtual Classroom with a Time Shift

by Luděk Matyska, Eva Hladká, and Petr Holub

8th International Conference on Information Technology Based Higher Education and Training. Kumamoto, Japan, July 2007. Proceedings. Kumamoto University, 2007. 6 p.

# Virtual Classroom with a Time Shift

**Luděk Matyska** [1,2] and **Eva Hladká** [1,2] and **Petr Holub** [1,2]

[1]CESNET z.s.p.o, Zikova 4, 162 00 Prague, Czech Republic
[2]Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic

## ABSTRACT

In spring 2007, Prof. Sterling's *Introduction to High Performance Computing* has been delivered in a virtual classroom consisting of 5 sites spanning Louisiana, Arkansas, and North Carolina in the USA and the Czech Republic in Europe. By utilizing a multi-point uncompressed audio and High Definition (HD) video capturing, transmission, and presentation system, a high fidelity collaborative environment was available to students, giving them access to the lecturer regardless of the place they attended the lecture. However, the semester start differs at individual participating educational institutions by several weeks and the institutions are also in different time zones. A store and replay technology was used in these cases, together with a larger number of lectures presented per week, to accommodate for this difference. The whole system worked well, the high quality of the HD video allowed students even in the most distant place to get a very good feeling of participating in a live lecture.

## INTRODUCTION

Providing high-quality university lectures on specialized advanced topics is increasingly difficult for smaller and less rich universities, as the top field specialist are very scarce resource. To mitigate this situation and to help their less developed peers, some top class universities are providing their lecture material, including captured video of individual lectures, free of charge to everybody interested. The MIT's OpenCourseWare activity is among the most widely known [1]. While helping to improve local teaching quality, this approach does not provide direct access to the teacher and keeps students at individual universities out of contact. Educationally more advanced (and more challenging) is thus the concept of a *virtual classroom*, where students at different places are synchronously attending the lecture. While the concept itself is rather old [1,2] and initially covered one campus only, the recent advances in network and audio/video technology open up access to a really high fidelity environment. The distance between individual halls of such a virtual classroom does not play serious role, students from differing institutions can share a particular lecturer, perceiving the illusion of being in direct contact with him or her.

[1] See http://ocw.mit.edu/index.html.

We have implemented a state of the art multi-point uncompressed HD video over IP transmission system together with other supporting technologies that can be used to provide such a high-fidelity environment. We have expected that our system would provide enough immersion for the students in order to feel being part of a single class, despite being divided by both geographical and time distances. As a pilot experiment, prof. Sterling at the Louisiana State University developed an *Introduction to High-Performance Computing* class that was offered to be shared in the virtual classroom setup at several universities and higher education institutions. The institutions participating in the class were Masaryk University (MU) in the Czech Republic, University of Arkansas (UARK), Louisiana Technical University (LATECH), MCNC and North Carolina State University (NCSU). In this paper, we present the necessary technical details of the setup together with a discussion of the experience on psychological and sociological levels.

From both technical and psychological points of view, the ultimate goal of a system to support true globe-spanning distributed class is to provide minimum disturbance for all the students of a class, while mitigating the separation of students on the remote locations and providing the same opportunities to both local and remote students. In order to achieve this, high-fidelity real-time audio and video transmission system have to be implemented, as further discussed in the following section. However, the real-time transmission and collaboration support is not sufficient, as the distributed classes face time-shift problems. One, simple problem is the time zone shift. While it may be possible to agree on a time that is convenient for two sides, with more sides and full globe coverage, not everybody is able to participate during his usual working hours. More serious problem is caused by differences in start of the class. Order of weeks skew may be found among individual institutions. To overcome both problems, the system must support non-synchronized, non-real time mode of work and the lack of real-time interaction should be taken care of at the educational process level, too.

## INTERACTIVE COLLABORATION TECHNOLOGIES

We decided to build our virtual classroom environment of top of the real-time collaboration environment using uncompressed audio and HD video and

over IP [3]. The HD video offers very high detail, allowing, e.g., to capture classical projection screen and to transmit the picture without any distortion (and loss of readability) to the remote location. It is also possible to capture both the projection screen and the lecturer. Using the HD video, students can see detailed facial expressions of the lecturer (and she can see the facial gestures of the remote students), providing a realistic remote presence. However, this high quality resolution does not come without a price—a lot of data is generated, which means the compression takes a long time even on high performance hardware. The several second long lag created by the compression and decompression of the video stream is unacceptable for the real-time collaborative environment. This lag can be easily removed using the uncompressed HD video.

The uncompressed HD video was HD-SDI over IP transmission based on UltraGrid software [3], featuring full 1080i SMPTE 274M/292M video with effective resolution of 1920×1080, 60 interlaced fields per second, 4:2:2 color space sampling, and 10 b per color plane.[2] The stream was packetized into the 8,500 B Jumbo Ethernet frames with 44 B IP/UDP/RTP header overhead and the resulting data rate was about 1.5 Gbps. With this data rate, we decided to distribute the video asymmetrically, in 1:N way from site where the class is given to all the other sites, and in N:1 way back, i.e. all sites are sending back to the lecturer only (this way the lecturer can see all his students but students can not see other lecture halls apart from the one with a lecturer). This asymmetry was implemented due to bandwidth constraints at participating sites.

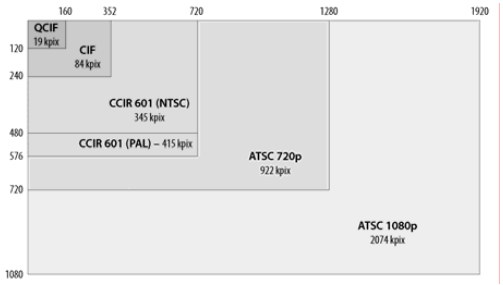



**Figure 1:** HD video formats

The audio part has been implemented by RAT [4] at 16 b quantization and up to 48 kHz sampling rate in stereo. Thus data rates up to 1.6 Mbps were used. The audio was distributed in full N:N way, i.e., all participants can hear and talk to one another. While the compression and decompression of the audio is not so computationally demanding as the processing of HD video, we used uncompressed audio to provide highest quality possible.

The high data rate needed for the uncompressed HD video streams transmission is not widely available on the shared production networks. We used a dedicated high speed network described in the following section, but we have also implemented a backup solution based on AccessGrid and webcasting in QuickTime format for those unable to participate at full network data rate.

## Network and Data Distribution Setup

The five partners participating in the virtual classroom needed a multi-point distribution of the video and audio streams. The very high data transmission rate disqualified use of native multicast—the routers usually do not support such high multicast distribution rate. We built an overlay network where specific nodes—the reflectors—took care of multi-point data distribution (see Figure 2). While a shared production networks available in academic environment already provide data rates above 1 Gbps at some locations, we used a dedicated 10 Gbps network as the underlying transmission medium.

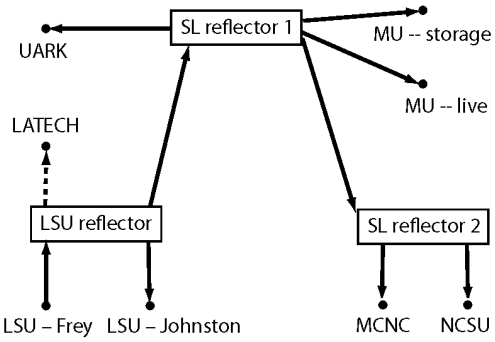



**Figure 2:** Overlay data distribution network for the 1.5 Gbps uncompressed HD video streams

The network has been implemented using experimental 10 Gigabit Ethernet (10GE) infrastructure and both statically and dynamically allocated λ-services. The network topology resembles star with center in StarLight (SL) in Chicago, IL. LSU and LATECH[3] were using 10GE backbone Louisiana Optical Network Initiative (LONI), which has an uplink to SL using a dedicated Enlightened wave running 10GE on National Lambda Rail (NLR). The Enlightened wave

[2] For more details on the HD formats see e.g. *Society of Motion Picture and Television Engineers:* "The Bit-serial Digital Interface or High Definition Television Systems", SMPTE-292M-1998 and "1920x1080 Scanning and Analog and Parallel Digital Interfaces for Multiple Picture Rates", SMPTE-274M-1998.

[3] LATECH is not part of the actual 10GE infrastructure though, because of the last-mile problems from the LONI point of presence at the LATECH campus to the room used for the teaching and thus they are using AccessGrid as the backup solution.

was being dynamically provisioned for the class and for the testing windows using a preliminary version of HARC [5], the software stack developed by the Enlightened project. UARK was connected via OneNet network and a wave on NLR running 10GE; again, the circuit was automatically setup before the class begins and shut down after it ends. MCNC and NCSU were connected to SL in the same way using NLR waves. MU, the only transatlantic partner, was connected using a dedicated permanent circuit Prague–SL leased by CESNET and the Brno–Prague implemented using CzechLight experimental infrastructure based on leased dark fiber lit by CESNET equipment. While the rest of the infrastructure runs 10GE, Brno–SL circuit runs OC192 protocol. The whole infrastructure is switched L2 network and is transparent on the IP level; only a few important network devices have been assigned IP addresses for monitoring and debugging purposes (see Figure 3).

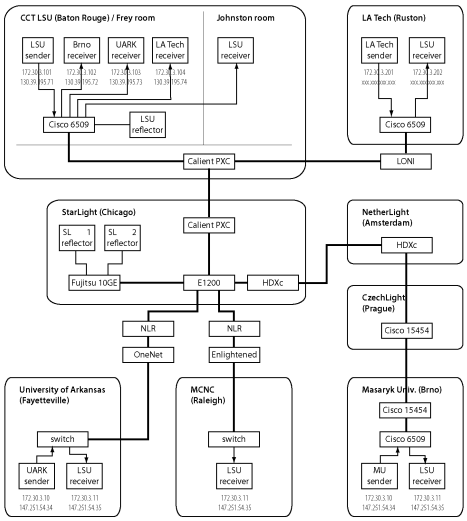



**Figure 3:** Detailed data distribution overlay network

Latencies are the major problem in real-time collaborative environments—with higher latencies it is not possible have a true dialog. The network latencies shown in Table 1 are sufficiently low not to create interactivity problems even in case of the transatlantic connection.

**Table 1:** Network round-trip latencies (RTTs) in the 10 Gigabit infrastructure

| Ping from | To StarLight | to LSU |
|---|---|---|
| LSU | 30.6 ms | - |
| MU | 115.4 ms | 145.7 ms |
| MCNC | 23.5 ms | 53.8 ms |
| UARK | 19.3 ms | 49.6 ms |

The actual data distribution was implemented using user-empowered software UDP packet reflectors [6,7]. The reflectors ran on dual-Opteron computers equipped with either Chelsio or Myrinet 10GE NICs. For the audio, although distributed in the full N:N pattern, a single reflector distributing is sufficient.
The video was distributed in 1:N and N:1 way and the schematics of the 1:N video distribution is shown in Figure 2. Note that there are actually two sites participating at LSU as not everybody can fit in a single room—thus the same technology was used for distributing the data locally on the campus between two buildings. There were also two streams being sent to MU—one of them was used for live video feed, while the other was used for the full-quality recording.

## Site Setup

Each site was equipped with one sender and one or more receivers for uncompressed HD video over IP transport system. The machines were essentially the same as used for the reference UltraGrid 1080i implementation [3], i.e., dual-Opteron computers with Myrinet or Chelsio 10GE NICs. The sender computers were equipped with DVS Centaurus HD-SDI capture card, while the receivers used NVidia graphics card to render the video on attached LCD screen or projector. 24" to 30" LCD screens and plasmas were used and installation at MU also used the Projection Design Cineo3+ 1080i projectors. The video part runs flawlessly with very good user perception.

Though less challenging from the networking and processing perspective, the audio part created many more problems due to various factors namely on audio capture side. The worst problems are caused by bad audio installations (ranging from wiring problems to echo canceling, gain control problems, and over-processing of the sound by various components before it gets to the computer—all this resulting in noises and distorted sound) used within the lecturing rooms primarily at LSU and wireless microphone interferences. Problems have also been encountered when using some on-board integrated sound cards, as their obviously half duplex behavior resulted in clicks and sound distortions. After appropriate corrective actions were undertaken, the sound quality has become very high and pleasing to hear.

## Data Storage

As stated above, the lecture had to be presented to students also in the asynchronous way. This was achieved through the recording of the live stream for a later use. The media streams (1 video and N audio streams) from the network were captured and stored as raw data on a disk array. Varying slightly depending on the size, each 1.5 hr lecture was about 1 TB of stored data.

The data storage has been implemented using the very fast local disk array comprising 12 disks (actually 6+6

disks attached to two physical channels of one SATA to SCSI disk array controller) in RAID 0, as it required both read and write throughput of at least 190 MBps. The actual write performance of the array was 385 MBps and read performance was 414 MBps. The tested parallel filesystem PVFS did not provide sufficient robustness with the inexpensive hardware we used. The stored streams were indexed so that seeking capability was also available. The data was archived to a slower RAID 5 disk array immediately after recording and later also to a tape archive so that there was always a backup copy available. When needed, the data was replayed from the same RAID 0 disk array.

## TIME-SHIFT MITIGATION

The time warping had to be implemented for two scales of time shifting—the short term, as the students may not be forced for instance to stay up to very late night due to different timezones, and the long term, because of different semester start dates. Both cases actually happened for the *Introduction to HPC* class.

The time difference between LSU and MU is 7 hours, which meant that the lecture starting at 3pm local time in Louisiana was received at 10pm at MU in Brno. The two hour lecture ended slightly before a midnight Czech Republic local time[4]. The time zone differences with other partners were at most one hour, which did not cause a real problem. In theory, the time zone problem could be solved by careful planning, but as we found, the LSU internal rules made it impossible to move the lecture to the more appropriate morning time and we had to account for this.

There is unfortunately no way how to directly overcome the second problem, differences in semester start. The only possible solution is to relax the real-time requirement and use recorded data instead of the live lecture. The semester at LSU started already in mid January, while the MU starts only after mid February. With two lectures per week this meant a delay of 10 lectures (full 15 hours of recorded material). To catch this delay students agreed to take up to twice the number of hours per week (usually there has been three instead two lectures). However, this created another problem with the problem sets and other prescribed homework.

**Other supporting material.** The students were also provided supplementary material like all the slides, additional reading, problem sets and homework, etc. The supporting material must be uncovered individually for each student's group, to follow the differing speed of presenting the recorded lectures. The major problem has been associated with the problem sets and the homework. Usually 4 to 6 weeks were available to solve a problem set and return the report. However, with the "faster than real time" lecture presentation the time available could be reduced, as some "future" lecture might provide hints for the problem set solution before they are actually submitted by students. We maintained supplementary materials in an archive local to each of the desynchronized institutions. The eventual shortening of time for homework has been agreed with students. In some cases, students were given new problem set before the previous was submitted—we checked that the "future" lectures are harmless, not containing any unwanted hints. All the homework as well as mid-term and final exams were evaluated locally.

The lectures at MU were replayed at the same days (Tuesday and Thursday) as at LSU, but at early evening hours. In the middle of semester (also due to the different holiday times and other reasons that lead to few than expected lectures at LSU) we were able to synchronize the MU and LSU classes. We still continued to present the recorded lectures at 7pm but invited students to stay longer and watch also the live lessons given by prof. Sterling or his colleagues.

Most of the time the whole system worked well, but at several occasions (3 times during the 13 week long semester) the live transmission has not been possible—either the transatlantic or other part of the path from LSU to Brno was broken and not operational. All the lectures were regularly recorded also at LSU and post-processed there. We used these post-processed lectures if the recording at MU was not available. The post-processing created another problem—usually it had taken one week before the material was ready for presentation. When MU and LSU were not synchronized, we had enough local material to present, but after the mid-term synchronization, there was no safety margin (the LSU has just one lesson ahead). Once we had to use a different lecture (recorded keynote from an international conference).

## LESSONS LEARNED

As we are expecting to repeat the experience next spring with substantially higher number of participants and also for everybody interested in similar educational setup, we summarize the major issues here. Two most important lessons are: (1) it is possible to run a lecture in HD quality at the virtual classroom setup even over transatlantic distances, and (2) it is not easy and needs a lot of man power.

The amount of necessary manual work surprised us. Part of it was due to a lot of homework and problem sets students were expected to work on during the semester. All the problem sets were prepared with the LSU resources available and we had to adjust our

[4] We may consider it irony that both time zones use the same abbreviation—CET. It denotes Central European Time in Europe and also Central Time in the USA.

computing environment to provide a similar setup to our students. Also, the teaching assistants at LSU had to learn how to share their experience (including the process of evaluation student's work) with the staff local at other sites. Without this, students would be graded differently.

To operate the whole system and especially the network setup took another portion of man power. As stated above, most lines were dynamically activated and de-activated, some of them through the use of experimental HARC software. All the transmissions were real-time and there was just a little time before the start of a lecture—the setup was usually initiated just half hour earlier, leaving very limited time for reaction to any problem encountered. We have learnt that when using experimental network, it is imperative to have a fall back solution available. An alternative capturing and presentation system must be available at all sites, having less stressing demands on the network and thus enabling use of a shared academic production networks instead of the dedicated experimental one. If more than the 5 sites are to be connected, much more automation is also needed, including a constant monitoring of the quality of network, drop rate (esp. for the audio) and other parameters that influence the quality of perception. Ideally, the system should be able to react immediately to network problems and reconfigure either the network or the collaborative environment itself on the fly.

The importance of good audio can not be overstated. While the full HD video quality provided details as expected, when a lower quality backup solution was used, it did not have tremendous negative impact. However, when the audio quality deteriorated, the lecture became completely useless. Even small noise, clicks and other artefacts negatively influence the student's focus and ability to follow the lecture (and learn from it). New microphones, audio cards and other equipment were tried during the first lectures to provide the best audio experience, but some problems repeated the whole semester. We are considering to study the studio quality audio systems to provide a flawless audio; it is worth even the higher price of studio systems.

When the semester starts at different time, it is very important to plan ahead how to mitigate it. We "learned by doing", adapting to the student's requirements and ability to follow the lectures (e.g., we did not know in advance whether students will be able to take more than the two lectures per week). Next time, we would like to synchronize with LSU as soon as possible, as it lessens the problem of material revealing and, most importantly, allow direct collaboration among students at different institutions. Without attending same lectures students have a little in common (they solve different problem sets, they have different experience from the last lecture seen etc.).

We also learned that the lecture must be prepared with the virtual classroom setup in mind. While Prof. Sterling prepared a lot of homework and other material, none of them directly encouraged remote student's collaboration. For the next semester, we would like to see also team homework, when students from different institutions will work together to solve a particular problem set. More interactivity between the lecturer and students is also very much needed. Students must be explicitly asked to participate, because even with the advanced technology the ability to ask question remotely is something that people do not know automatically and it must be learnt. Encouraging direct interaction between students and asking for feedback during or after lectures will teach students how to live well in a virtual classroom environment.

## CONCLUSIONS

The collaborative system described in this paper had been implemented by the beginning of year 2007 and has been used since then for supporting the HPC class. Due to running on highly experimental networking infrastructure, some sites had to resort to using backup solutions at times, but the infrastructure worked flawlessly most of the time. Another experience gathered from this class is how important is high-quality audio installation at participating sites and how difficult it is to implement it properly. The low sound quality is much less tolerated when other means of communication are of high quality compared to common point-to-point communication tools.

We are currently undertaking a number of studies on subjective user perceptions in order to map the technology improvements to user perception improvements. As more sites want to participate, the whole system needs to get revamped in order to support much higher degree of monitoring and self organization.

## RELATED WORK

We are not aware of any e-learning system that uses the uncompressed HD video for real time lecture delivery. The virtual classroom concept is widely used, but with the more classical collaboration technologies like H.323 videoconferencing systems and simple shared work space.

Interest in HD video is growing, also due to the increased availability of adequate equipment designated for the consumer market. A recent presentation at EUNIS conference [8] gives a good introduction to the use of commercially available products that use compression and introduce thus latencies unacceptable for the live collaborative environments of the real-time virtual classrooms. The HD video is currently used for lecture streaming only.

## ACKNOWLEDGMENT

This project has been kindly supported by the research intent "Optical Network of National Research and Its New Applications" (MSM 6383917201). We would also like to thank other people working with us on this project, namely Tomáš Rebok, Miloš Liška, Lukáš Hejtmánek from Laboratory of Advanced Networking Technologies at Masaryk University, and our partners at Center for Computational Technology at Louisiana State University and other partnering sites.

## REFERENCES

[1] S.R. Hiltz, "The "virtual classroom": Using Computer-mediated Communication for University Teaching", Blackwell Synergy, 1986

[2] S.R. Hilts, "The virtual classroom: learning without limits via computer networks", Ablex Publishing Corp. Norwood, NJ, USA, 1994

[3] P. Holub, L. Matyska, M. Liška, L. Hejtmánek, J. Denemark, T. Rebook, A. Hutanu, R. Paruchuri, J. Radil, E. Hladká, "High Definition Multimedia for Multiparty Low-latency Interactive Communication", Future Generation Computer Systems, Vol. 22(8), pp. 856–861, 2006

[4] V. Hardman, A. Sasse, M. Handley, A. Watson, "Reliable audio for use over the Internet", Proceedings of INET'95, Honolulu, Hawai, 1995

[5] L. Battestelli, A. Hutanu. G. Karmous-Edwards, D.S. Katz, J. MacLaren, J. Mambretti, J.J. Moore, S.-J. Park, H.G. Perros, K.S. Sundar, S. Tanwir, S.R.Thorpe, Y. Xin, "EnLIGHTened computing: An architecture for co-scheduling and co-allocating network, compute, and other grid resources for high-end applications", 2007. Available from http://enlightenedcomputing.org/index.php?n=Main.AboutEnLIGHTenedPage?action=dowload&upname=EnlightenedGrid07.pdf

[6] E. Hladká, P. Holub, J.Denemark, "An Active Network Architecture: Distributed Computer of Transport Medium", Proceedings of 3rd International Conference on Networking (ICN'04), pp. 338–343, Gosier, Guadeloupe, 2004

[7] P. Holub, E. Hladká, L. Matyska, "Scalability and Robustness of Virtual Multicast for Synchronous Multimedia Distribution", Proceedings of 4th International Conference on Networking, La Réunion, LNCS vol. 3431, pp. 876–883, Springer-Verlag Heidelberg, 2005

[8] J. Oxenford, "Using HD Videoconferencing for Teaching and Learning", Proc. EUNIS 2007 Conference, Grenoble, Paris, 6 pages, 2007

M