AMnet: Efficient and Flexible Provision of Heterogeneous Multicast Services

Anke Speer, Ralph Wittmann, Martina Zitterbart
Technical University Braunschweig
{speer|wittmann|zit}@ibr.cs.tu-bs.de

Bernard Metzler
Technical University Berlin
metzler@prz.tu-berlin.de

Abstract

AMnet provides a framework for flexible and rapid service creation. Its primary goal is to provide customized multicast services on demand, thus providing a framework for a flexible open communication platform. AMnet is based on active and programmable networking technologies and uses active nodes (AMnodes) within the network. These AMnodes are executing on-demand loadable service modules to enhance the functionality of intermediate systems without the need of long global standardization processes.

This paper gives an overview on fundamental design choices for AMnet including the AMnet service concept. The architecture of an AMnode is discussed in some detail.

Introduction

In the past decades a tremendous growth in the use of computer networks in general and the Internet in particular can be noticed. Besides the provided interpersonal communication, like E-mail, the networks advance in becoming the media for distributed computation, tele-collaboration, distance learning, e-commerce and the like. Many of these applications are inherently based on group communication. Therefore, efficient and scalable group services are needed for proper communication support. The particular group service to be provided highly depends on the application itself and on the media transported, e.g., text, audio or video. Issues that vary within these group services are reliability and quality of service support. It is important to note that no single group service will be able to serve the high variety of existing and emerging group applications.

Heterogeneous demands on service quality and media format as well as dissimilar network conditions may lead to situations, where different group members would like to experience a different group service. Currently diverse approaches exist to signal and provide those heterogeneous group services [1, 2, 4]. However, the provided service is typically penetrated by the member with the lowest service capabilities. This is not acceptable for multimedia and collaborative applications in heterogeneous networking environments.

The AMNet approach makes user-tailored data streams available to individual receivers and, thus, enables heterogeneous group services [12]. Following the concepts of Active and Programmable Networks [17, 18], additional functionality can be placed flexibly within the network without the need of global standardization processes. Dependent on actual user requirements, appropriate service modules are located dynamically on active network nodes (AMnodes). These service modules then individually tailor data streams according to service specific requirements. Examples of such service modules are QoS-filter and modules for individual error control. Figure 1 shows the application of the AMnet approach to a typical tele-collaboration...
scenario, where the participants differ in their Quality of Service (QoS) requirements. AMnodes perform the adaptation of the original data stream in accordance with the desired QoS-levels.

It should be highlighted that AMnet represents a general framework for heterogeneous group communication. It is not limited to the application of specific service modules. AMnet rather provides an open architecture where new services can easily be incorporated.

The paper is organized as follows. The next section gives a brief introduction into the AMnet concept for the provision of heterogeneous multicast services. Section 2 presents the main building blocks for the management of AMnet services. Section 3 discusses the AMnode architecture. The paper closes with a summary and an outlook on ongoing research.

1 AMnet

AMnet aims at the provision of scalable heterogeneous group communication with efficient and rapid service creation. It is based on the placement of additional functionality inside the network. Service modules are responsible for the adaptation of data streams to specific service demands. These modules are dynamically loaded by Active Multicasting Nodes (AMnodes) which form the core building blocks of AMnet. AMnodes operate on the communication path between sender and receivers. In this sense AMnet represents a Programmable Networking approach.

A goal of AMnet is to support service heterogeneity transparently to the origin of a data stream as well as to the receivers. Therefore, AMnet service establishment follows a receiver oriented approach. Instead of capsules, loadable service modules with out-of-band signalling are used for service creation, since (1) capsules would have to be provided by the sender, i.e., the desired sender transparency and receiver orientation could not be achieved, (2) for performance improvement of AMnet service provision a combined HW/SW solution is envisioned. Both software based service module execution and programmable hardware support for dedicated service modules will be available. The structure of the hardware dependend modules significantly differs from software based modules, a realization as capsules would be complicated.

For the AMnet concept of loadable service modules an active node design was developed and implemented as a prototype. The reason for the development of an own implementation...
can be found in the lack of simple, ready-to-use experimentation platforms for active network services. Available systems are either using proprietary operating systems (e.g., [8, 14, 13]), rely on significant extensions of common operating systems to provide highly specialized service execution environments within kernel context [5] or, at least, use proprietary programming languages [7]. Further, the AMnode design was necessary to allow for the integration of both software and hardware based service provision.

2 AMnet Service Concept

An *AMnet session* or *session* defines a communication scenario where a designated sender issues a data stream which can be received from several communication participants after special adaptations. The necessary placement of application-dedicated functionality within the network raises some questions: where should those functionalities be located, how should corresponding services be established and maintained, how should a receiver be associated to a dedicated service and how should different services be managed within a session? This section briefly outlines the AMnet specific solutions for these problems.

2.1 Service Levels

Service heterogeneity within a session needs to be bound to a manageable degree of diversity. Therefore, one concept of AMnet service control is to logically group receivers with similar service demands into distinct multicast receiver groups – the *service level groups* (see Figure 2). Service level groups are distinguishable by their multicast addresses. Receivers join the corresponding group on demand through IGMP [6].

![Figure 2: Example multicast tree](image)

Each service level group within a communication session represents all receivers whose service demands can be resolved with a single group service. Therefore, each group represents a different view onto the same original data corresponding to the adaptation within the AMnodes. Receiver groups are hierarchically ordered. AMnodes are member of both, the parent service level group and the child service level group. For example, AMnode 4 in Figure 2 is member of the parent service level group of AMnode 1 and of its own child service level group. The data stream received from the parent may be an already adapted stream or the original stream. The
The scope of an AMnet service group is limited by the actual TTL value assigned to packets issued by the corresponding AMnode (see Figure 2). The TTL value of every group must not exceed the scope of the service announcement group (see below).

The establishment of service level groups permits the provision of different service qualities within one region without the services interfering each other. Data streams with different media formats or individual error control must sometimes coexist on a communication link and have to be distinguishable by the appropriate receivers. All packets are explicitly assigned to corresponding service level groups by their multicast destination addresses.

The hierarchical order of service level groups allows for an efficient establishment of different quality levels within a session. One distinct service quality might be easily derived from another already available quality level, if, for instance, only a different (weaker) error control policy for network overload conditions has to be inserted into the already adapted data stream. However, the hierarchical ordering does not automatically imply hierarchical degradation of all service parameters. Some parameters can be provided unchanged, other parameters even improve. As an example, the insertion of a new service can improve media playback quality due to less jitter at the cost of higher, but uncritical delay. This could be useful for video distribution, for example.

The service quality experienced at the receiver is a function of the service level of the group and the current network conditions between group source and each individual receiver. Thus, the service within a service level group can only differ in performance-oriented, packet-based service parameters such as offered delay or loss probability. Other parameters such as the content-based nature of the service (e.g., media format) or communication protocol mechanisms (e.g., acknowledgement or error control strategies) must be homogeneous. However, the distinction of services can be triggered by all parameters. Consider two receivers requesting the same media format but experiencing very different loss probabilities. In that case different acknowledgment strategies and error control mechanisms might be necessary which require distinct service level groups.

### 2.2 Service Location and Announcement

In contrast to the capsule approach, AMnet’s service creation is not based on executable code to be transmitted in data packets. Instead, the AMnet signalling provides procedures to locate and announce service modules and to establish and maintain appropriate services on the AMnodes. These signalling procedures utilize active networking technologies and deploy the group concept of the Internet as follows. For the management of services different groups are used: session control group, service announcement group and service level group (see Figure 3) [12, 10]. The control of the heterogeneous group services is maintained completely out-of-band. If a receiver joins these groups, it receives the data packets exchanged there. For AMnet participation, the receiver has to join the session control group. On the basis of the received session announcements the receiver decides whether to join a session. In the case of participation, the receiver joins the session’s service announcement group to learn about available services for this session. If the receiver wants to use an AMnet service to adapt the original data stream to an appropriate quality level, it examines the service announcements whether the desired service is already provided by an AMnode. If already available, the receiver joins the appropriate service level group of the selected AMnode. The participation can depend for example on metrics like distance or workload of the groups [16]. If the service is not available, a search in the multicast tree is started. Then, the receiver asks an – according to the propagated metrics selected – AMnode to provide the service. The AMnode has to load the appropriate service module out of the service module repository and has to announce its establishment in the service announcement group (see Figure 3).
In order to evaluate the developed signalling protocol for AMnet, first simulation experiments were carried out using the simulator ns-2 [16]. Issues of interest throughout these simulations include the frequency of AMnode setups and the delay between service request and provision. The simulations examined the influence of variable configuration parameters, e.g., the lifetime of a service module on an AMnode, the time for loading and installation of service modules on AMnodes and the total number of AMnodes in the network. The allowed distance between receiver and providing AMnode and the lifetime of the service modules were found to have a positive effect on the time a receiver has to wait for service provision as well as on the needed signalling effort. To put it in a nutshell, the developed signalling protocol is considered to be appropriate for the flexible and dynamic service creation AMnet aims at.

3 AMnode Architecture

Research on Active Networking has recognized the importance of active node design for service flexibility, security and its performance. An “architectural framework for active networks” introduces the main components of active nodes which are expected to be common to all node realizations [3]: an Execution Environment implements a virtual machine which can be programmed in-band or out-of-band. Thus, the way packets are processed at active nodes becomes controllable. An active node may host multiple Execution Environments simultaneously, where within each environment the execution for one or more different services may take place. The NodeOS implements functionality to mediate between the nodes’ physical resources (transmission bandwidth, CPU time, local memory) and its use by active Execution Environments. Another common functionality of the NodeOS is the demultiplexing of incoming packets to assign them to appropriate instances providing flow specific services.

The AMnode architecture was designed to support the execution of service modules on a commodity operating system. A first prototype was implemented on HP9000 computers running HP-UX. It is being ported to the Linux OS. The realized architecture can be explained using the introduced terms of an Execution Environment for flexible services and an NodeOS (see figure 4). Most parts of the Execution Environment are realized at application level. Moderate kernel extensions provide the necessary NodeOS functionality. They are responsible for a proper local resource management and for the support of safe and efficient data movement.
between Execution Environment and network interfaces. For efficiency reasons, some parts of service specific packet processing can be delegated to kernel level processing (the kernel packet processor) or even offloaded onto dedicated hardware. The node architecture is discussed in more detail below.

![Figure 4: Overall AMnode Architecture](image)

### 3.1 The NodeOS

The NodeOS consists of a packet classification engine, a UNIX kernel device, programmable hardware and a user-level network access demon.

**Packet Classification** One major task of the kernel extension is the correct demultiplexing of incoming data to corresponding flow specific service modules at user level. A received packet should bypass ”normal” kernel networking, if an AMnet service is registered for it. The packet classification is based on the well-known Berkeley Packet Filter (BPF) [9]. It is extended for more efficient demultiplexing on a large number of active data flows. For 25 endpoints, the BPF/C (Berkeley Packet Filter/Classifier) shows a mean demultiplexing delay of about 16 µs and is, thus, more than three times faster than the BPF (see figure 5(a)). Each point represents the arithmetic mean of 500 packets processed on a HP9000/735-99 workstation. Efficiency improvement is achieved due to the dynamic merging of filtering instructions from different associated flows into one filtering instance. If the BPF needs \( n \) instances to demultiplex \( n \) flows, the BPF/C is executed only once and, thus, acts as a packet classifier. As a further extension to the BPF, the BPF/C supports the correct classification of segmented packets. A more detailed discussion of the BPF/C-approach can be found in [11].

The BPF/C packet classifier is also used to enforce correct header syntax of outgoing, application-level generated data. Since outgoing packets are generated in a potentially untrusted environment, this module avoids the insertion of malicious packets into the network. Packets not matching the appropriate protocol syntax are refused.

**The ltap Kernel Device** The kernel extension is responsible for the transfer of data between network adapter and application-level execution environment. The necessary functionality is provided by a UNIX device driver called *ltap* (link tap). The semantic of its usage is similar
to the socket abstraction. For each active AMnet flow the /dev/ltap file has to be opened and assigned with appropriate addressing information to send and receive packets. Internally, for each endpoint send and receive queues are established. For the enforcement of node resource management to each queue a specific watermark is assigned. Exceeding this watermark on the receive path causes packet dropping with a pre-selected dropping policy (head or tail dropping). Dependent on selected API semantic, an overfull send queue may result in a blocking write or an appropriate error code on syscall return. The prototype AMnode offers an encouraging performance. As an example, on the same hardware as mentioned above the mean input delay for an UDP/IP packet of 1 KByte is below 80 $\mu$s. This is significantly better than the obtained delay of about 110 $\mu$s on a standard UDP/IP socket (see figure 5(b))\(^1\).

To enhance the performance both on input and output path a single copy operation is selectable. Data copy operation on the input path can be avoided by application-transparent page remapping between kernel and application. On the output path, the ltap device driver offers single copy operation due to direct data movement between application address space and I/O address space – for a detailed discussion see [11].

The system call overhead on packet input remains an open issue, since a substantial redesign of process scheduling mechanisms was avoided. Nevertheless, within the given architecture it can be significantly reduced for large application level frames, if the optional kernel packet processor is applied to collect all frame fragments before presenting the data to the execution environment (see 3.2).

**Network Access Demon** Although running as a priviledged application-level demon process, the *Network Access Demon* (NAD) is conceptual part of the NodeOS. The NAD is responsible for the correct assignment of filtering instructions, for packet classification and output packet syntax check. During AMnet flow setup, the requesting AMnet service entity attempts to bind the desired address information to the open ltap device. This request is delegated from the ltap device driver to the NAD. If the address is not already in use, the appropriate packet classification information is installed. The NAD is responsible for the generation of efficient BPF/C code. Its realization as a user level process allows for the easy extension of the NodeOS to support various packet formats.

**Programmable Hardware** AMnet allows for the offloading of service module execution onto programmable hardware. Both DSP-based and FPGA-based solutions are supported. The NodeOS is responsible for the installation of requested hardware based service modules (*happlets*) and for the data transfer between execution environment and hardware boards.

\(^{1}\)To obtain comparable results, the in-kernel UDP packet checksumming was temporary switched off.
realized solution allows for the transparent integration of hardware and software based service module execution. [12] gives a more detailed discussion on the chosen design.

3.2 The AMnet Execution Environment

The AMnet Execution Environment implements the AMnet specific part of an Amnode. It consists of functional modules to implement AMnet signalling, AMnet service management, QoS monitoring and AMnet service execution.

The local AMnet signalling is responsible for the announcement of local service capabilities, for handling incoming service requests and retrieving of service modules not locally available (see chapter 2.2). It interacts with the local AMnet service management which installs and maintains AMnet service modules to create AMnet services. Interacting with the QoS Monitor the service configuration is complemented with local resource management. After service establishment, the QoS Monitor collects actual information on ongoing service provision. This information is intended to be used for service announcement and service admission control.

Service Environment Composable services are constructed from components which reside within the Execution Environment [19]. For AMnet, the level of component granularity is the service module. Each service module implements a well defined part of a given AMnet service. Beside service specific functionality such as media transcoding or flexible error control also basic communication protocol tasks such as network level packet de-/fragmentation, checksumming and packet header generation are performed within service modules.

The Service Environment is created under control of the AMnet service management entity. It is also responsible for the loading of service modules into the established environment and service configuration (see below). The Service Environment offers configuration support, memory and timer management (see figure 4). Within a Service Environment, one or more services can be provided. Furthermore, the design of the Execution Environment allows for the coexistence of more than one Service Environment, where each environment is realized as a separate application process. This allows for the easy integration of service modules which may block during execution (such as hardware-supported modules - see [12]), where multithreading would be necessary otherwise. Further, node-local resource consumption can be controlled on a per process level.

Service Configuration and Execution An AMnet service is composed from so called Activities which have to be performed on the occurrence of Events. An Activity is defined by the sequence of service modules to be executed and their actual parameters. More formally, it is a directed, acyclic graph connecting a subset of installed service modules and parametrizing them dynamically. Events can be raised on packet reception, local timer expiration and from already active service modules (this implements branching Activities).

The configuration information is given to the local configuration management. It builds an Activity Path, where each service module is represented by its function pointer and associated calling parameters. Further, the structure integrates an interface to hold and module specific reference associated data (typically network packet structures) which has to be manipulated during Activity execution. After setup of the Activity Path the configuration management returns a unique handle to the caller.

If an Event occurs, the appropriate handle is presented to the configuration management. It responds with a copy of the associated Activity Path. Often during an Activity a packet has to be processed (e.g., packet reception from network). In that case the caller has to concatenate the packet with this control structure. Now from the Activity Path the first service module is determined and called by dereferencing its pointer. The only argument which is explicitly
given to the service module is the control structure itself. Successive calling of following service modules is done under control of the currently active module by simply dereferencing the following service module pointer from the control structure and, again, handing over only that structure. For a consistent path execution, per definition (1) each AMnet service module has to call the next module as its last operation and (2) if called, each module has to expect a pointer to the Activity Path as the only explicit parameter.

That rather simple management of control flow during service execution allows for good efficiency. After Event reception and simple Activity Path initialization any interaction with a control instance is avoided. Further, holding state informations of any ongoing Activity within a separate structure allows for the simple coexistence of Activities within a service environment and the dynamic reconfiguration of an Activity Path without affecting its possibly ongoing execution.

**Kernel Packet Processor** An optional kernel level packet processor introduces the possibility to flexibly download application specific code into the kernel. Each AMnet flow is permitted to optionally install one input and one output processor. The programming language of the packet processor is derived from the BPF and extended for the expression of desired functionality. It is currently restricted to simple packet parsing, splitting, concatenation and duplication as well as packet sending operations. Sanity checks on application-originated programs are forced to prevent the insertion of malicious code into the kernel. On the input path a typical use of the kernel packet processor would be the collection of all fragments of a higher level frame before its presentation to the AMnet service modules. Such it supports for the execution of all modules necessary for a given AMnet service without intermediate path termination due to still incomplete frames. This helps to minimize service environment scheduling.

## 4 Summary and Outlook

In this paper an outline of the AMnet approach for flexible and rapid service creation in IP based networks was presented. Programmable Networking technologies are used for service creation through loadable service modules. This mechanism helps to enhance the functionality of the network and the use of Active Networking technologies provides this enhancement without long global standardization processes. The design of an active intermediate system for efficient and flexible AMnet service execution was introduced.

Future work includes the further development of the AMnet signalling protocol. Here, the focus is on the introduction of merging and reconfiguration mechanisms for service modules and their localization inside the network. A testbed for AMnet will be created. Moreover, the integration of AMnet signalling and the AMnet Execution Environment is carried on.

## References


