Results on the Practical Feasibility of Programmable Network Services

Thomas Fuhrmann, Till Harbaum, Panos Kassianidis, Marcus Schöller, and Martina Zitterbart
Institut für Telematik, Universität Karlsruhe (TH), Germany

***Abstract*—Active and programmable networks have been subject to intensive and successful research activities during the last couple of years. Many ideas and concepts have been pursued. However, only a few prototype implementations that have been developed so far, can deal with different applications in a larger scale setting. Moreover, detailed performance analyses of such prototypes are greatly missing today.**

Therefore, this paper does not present yet another architecture for active and programmable networks. In contrast, it rather focuses on the performance evaluation of the so-called AMnet approach that has already been presented previously [1]. As such, the paper demonstrates that an operational high-performance programmable network system with AAA (authentication, authorization, and accounting) security functionality will in fact be feasible in the near future.

I. Introduction

Active and programmable networking research has produced many promising ideas during the recent years. Now, the success of these concepts and laboratory prototypes depends on their practical feasibility. Which services yield an obvious benefit to the end-users and network providers so that they are willing to use active services? How can these services be dynamically provided by and for the different stakeholders? How can authentication, authorization, and accounting be achieved? And, maybe most important, does the resulting overall system have enough performance to be of actual practical use?

In this paper we present first results from the performance evaluation of sample services that have been implemented within the AMnet programmable networking platform used in the FlexiNet project. The AMnet platform comprises not only the active nodes themselves but also a hierarchical repository structure and network management facilities. This platform is briefly described in section II.

The sample services that we studied in our performance measurements were chosen for the direct benefit that these services can give to service providers and end-users. They are described in section III. We strongly believe, that their capability to produce direct and obvious benefit for users and providers will be of utmost importance to the successful deployment of active and programmable networking technology.

Section IV presents first results of our measurements. Generally, these results are very promising and indicate that active and programmable networks will in fact be feasible in the near future. Finally, section V concludes with an outlook to future work that needs to be done before programmable networks can actually be deployed in the large scale.

Please note that, for sake of brevity, we refrain from discussing AMnet in relation to other approaches. Please confer to [1] for an overview of similar approaches that places AMnet in its context.

II. The AMnet platform

The AMnet approach[1] has been developed over the last couple of years. Recently it has been part of the *FlexiNet* project[2] which investigates the applicability of programmable and active networks for various kinds of services and support issues. The project involves the Technical University of Berlin, the University of Karlsruhe as well as the Technical University of Munich. Karlsruhe basically provides the programmable platform whereas all three universities contribute applications that can benefit from such a platform. In this context, various applications have been developed, including support for heterogeneous receiver groups, security enforcement and mobility support.

The project proceeds iteratively, pursuing two interwoven aims, namely, on the one hand to provide a programmable network testbed that allows the testing of flexible network services and, at the other hand, to also develop such services. The latter then creates the insights required to improve the former.

[1] AMnet originally meant *active multicasting network*. Meanwhile, however, multicast is only one out of many possibilities for signaling in AMnet.

[2] Funded by the German Ministry of Education and Research BMBF under grant number 01AK019A

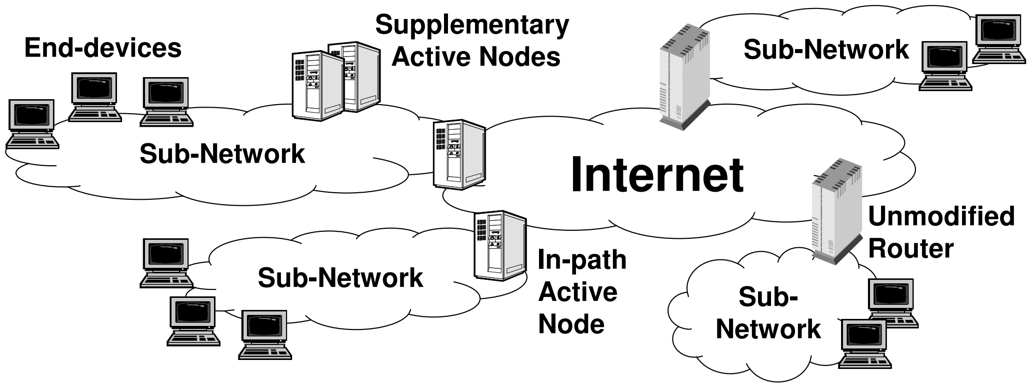

Fig. 1. Architectural overview of the FlexiNet system: Active nodes are placed in the edge domains either as border routers or as supplementary nodes off the data path.

In the following, we briefly describe the key design features of the AMnet platform.

The AMnet platform is centered around the concept of *service modules*. These are small pieces of code that can be drawn from administratively managed *service module repositories* (SMR) that also provide the respective authentication, authorization, and accounting functionality. The code may come in different flavors to be executable on different hardware platforms. Corresponding parameters in the associated database of the SMR determine the respective module's resource requirements in terms of hardware components, memory, bandwidth, access rights, etc.

Service modules either provide a service by themselves or they are combined to create a more elaborate service. Breaking down a complex service into simpler service modules is a powerful means for code reuse, too. So far, no formalized mechanisms have been implemented to describe the functional effects of services modules, although concepts for that are currently being developed in the FlexiNet project. It is rather up to the respective service implementor which service modules she wants to create and which existing modules she wants to combine with them to achieve the intended purpose. Nevertheless, a globally unique naming scheme ensures that services are always correctly assembled from the service modules when they are executed on an active node.

All FlexiNet services can be categorized into two major types of services, *application services* and *infrastructure services*.

Application services are initiated (and potentially controlled) by applications running on the end-devices. Thus, these applications need to interact with the AMnet signaling. Either, an application is *AMnet-aware* by itself, or, if, e.g., an existing application cannot be modified to become AMnet-aware, an additional control application can be employed to setup the desired service. In any case, it is more a less the user's decision to use the programmable network functionality. Either she uses a service implicitly by using an AMnet-aware application, or she uses it explicitly by invoking the service manually from a control application. Accordingly, users will typically be charged for using a service.

Infrastructure services, on the other hand, are launched by the network administrator. Such a service can run quietly and become active only upon certain conditions in the network, e.g., an intrusion detection service, or a service that can block distributed denial of service attacks. Other services run permanently, e.g., an optimized last-hop protocol for wireless access networks. Unlike application services, infrastructure services are not charged to the user but create value by improving the security, reliability, or performance of the provider's network.

Both categories of services are executed on programmable active nodes, the so-called *AMnodes*. These nodes should be routers or dedicated systems that are located somewhere near the edge of the network. Typical places would be an access router serving a home or small business network, a router serving a floor in an office building, etc. (cf. figure 1). The AMnodes are definitely not designed to be placed in the backbone of network. Besides being placed as a router in the data path, AMnodes can be placed off-path, too. There, they assist the in-path nodes with sharing the processing load generated by resource intensive services.

AMnet provide the complete framework required by an active node: The *execution environment* allows services to inspect and manipulate (certain parts of) the network traffic. The *resource control* manages the operation of the active node itself, checks that services do not exceed their granted resources, and ensures the node's integrity by avoiding overload situations. The *signaling* engine

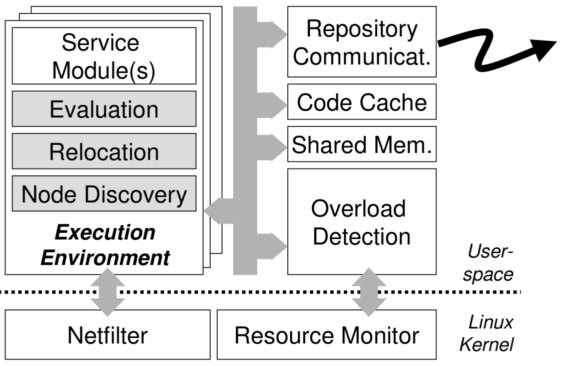

Fig. 2. Architecture overview of an AMnet active node

controls the communication both with applications using the active network and the service module repository controlling the use of the active network.

The AMnode framework also provides basic functionalities to determine an AMnode that is well-suited to execute a requested service (*evaluation*), or to relocate an already running service (*relocation*) when either the users' requirements change, e.g. due to their mobility, or for load balancing reasons (see also figure 2). The detailed explanation of these mechanisms is, however, beyond the scope of this paper.

III. Sample Services

Compared to the multitude of interesting conceptual approaches in active networking, only very few publications discuss the practical issues that come up with employment of real-world active applications with active and programmable networking platforms. As explained previously, the FlexiNet project is explicitly designed to also investigate these practical issues.

In the previous section it was described that the AMnet platform incorporates many important architectural features like AAA-functionality, resource control, load-balancing, etc. that form a sound basis for the deployment of real-world active applications in a programmable network. This section is concerned with examples for such real-world active services. Such examples are necessary to evaluate the feasibility of active networking.

During the course of our research, we have already implemented several services for the AMnet platform that we believe to come close to actual real-world services, i.e. they largely exceed the 'active ping' that was, e.g., used in the ABone test network. We do not claim any of the applications to be new. In contrast, many of them have already been discussed in the literature. However, to the best of our knowledge, none of the approaches described here has been intensively considered with regard to the performance that can be achieved in a mid-size programmable network.

In this paper, we describe the performance evaluation of three sample services which we consider very instructive and helpful to convince the potential beneficiaries of active networking, namely the Internet Service Providers and the end-users:

1) Application layer multicast reflector. This reduces the required uplink bandwidth when several clients receive the same content. We demonstrate this with an MP3 Internet radio server.
2) HTTP compression. This service is especially suited for mobile end-devices with low-bandwidth connections.
3) Video transcoding. Videos are transcoded according to the requirements of a handheld device, in our example set-up, a Palm-OS Personal Digital Assistant (PDA).

The first two of these services operate completely transparently to both, the sender and the receiver. The Palm PDA example uses a special viewer application that is capable of controlling the FlexiNet service parameters, e.g. desired quality, black-and-white versus color, etc. Nevertheless, the service is still transparent to the sender.

More detailedly, the multicast reflector intercepts a TCP stream that carries an HTTP request to and the response from an Internet radio server. (Currently, we use a predefined list of IP addresses of popular servers to identify the stream.) The MP3 payload of this connection is both, forwarded to the MP3 receiver and stored in a ring buffer. If a second client requests the same URL, the connection is terminated on the active node and the ring buffer content send to both clients. The same applies to all subsequent connections. If the first client terminates its connection, this connection is maintained upstream between the AMnode and the server, so that the buffer can still be filled with data. The server connection is closed when the last client connection has been closed.

The HTTP compressor uses the same TCP interception mechanism. However, this time, the content is not stored in a buffer but compressed to one of the formats supported by the web-browser. (This is indicated in the HTTP-request.) The Palm video transcoding service, finally, transcodes MPEG2 data to a specialized format that a Palm PDA can display more easily. The motion prediction and digital cosine transformation have to be decoded by the AMnode since the Palm has not sufficient computing resources to do these steps by itself. Furthermore, the color data is specifically matched to the Palm's color palette. So, the PDA can directly write the received data into the video display memory without the need of any

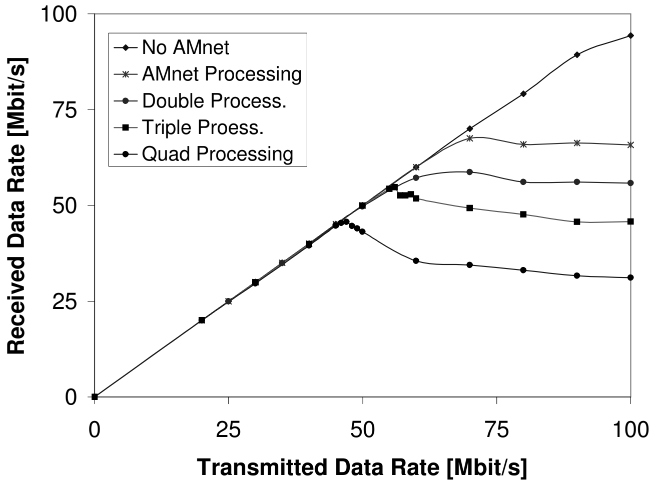

Fig. 3. Performance of a 800MHz Pentium active node (100 MBit/s Ethernet, 1460 byte packets): Throughput drops from 92 MBit/s to 66 MBit/s when the service is switched on.

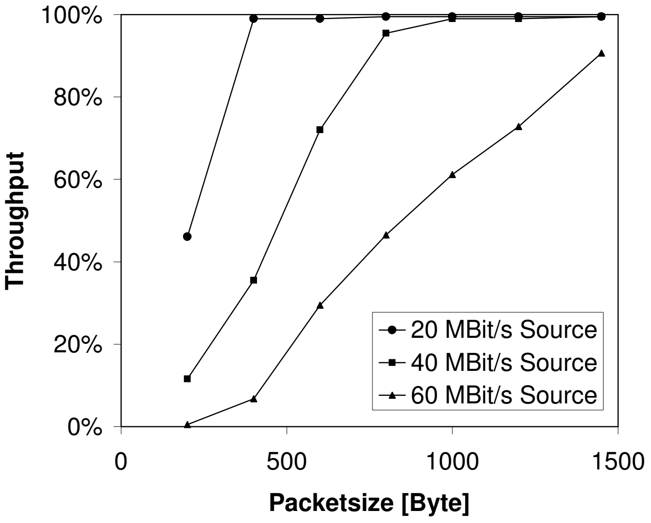

Fig. 4. Throughput depending on packet size: Only large packets exploit the theoretically achievable throughput. (Same set-up as fig. 3)

complex operation.

Device-specific services are especially illustrative example for the use of programmable networking technology. The following section will show that such services are in fact practically feasible.

IV. Performance Evaluation

After having briefly presented both, the AMnet platform and three sample services that have been implemented for this platform during the course of the FlexiNet project, we can turn to the main focus of this paper, namely to the performance evaluation of such a programmable networking system.

Figure 3 shows the result of a measurement whose purpose was the demonstration of the achievable throughput on an ANnode. This measurement is important for services like the MP3-multicast reflector that perform only marginal operations on the packet (duplication, address substitutions). The measurement set-up employed three machines, a sender, an active node, and a receiver that were put in line, connected with a dedicated 100 Mbit/s Ethernet. The active node was a 800MHz Pentium machine. It ran a simple benchmark service that performs a certain number of processing steps for each byte of each packet. Packet size for these measurements was 1460 bytes.

The measurement results presented in figure 3 demonstrate that our active node is capable of a high throughput, i.e. we can achieve a throughput of 68 Mbit/s in a scenario where a 100 Mbit/s Ethernet was used as underlying network. If we double, triple, or quadruple the processing load per packet, the maximum throughput drops to about 30 MBit/s.

Obviously, there is a constant decrease in throughput for each time the service is executing the processing loop, namely, for this example, about 12 MBit/s per processing loop. But this decrease is significantly smaller than the initial decrease when the service is activated (92 MBit/s to 66 MBit/s, i.e. 26 MBit/s). From this, we can conclude that our active node creates a certain base load, equivalent to 14 MBit/s decrease (26 MBit/s minus the 12 MBit/s that can be attributed to the processing of the packet within the service). Hence, for more complex services the main load is created by the service itself, i.e. the active node must provide enough processing resources to keep pace with the incoming traffic. This is especially important for transcoding applications (see below).

Observing that the base load creates about the same as the simple packet processing, we can conclude that this is the overhead created by transferring the packets into the user space. Although, this seems to be wasted performance, we believe the advantages of running the service in user space worth the extra effort that cost about 15% of the node's performance.

Another interesting feature is the existence of a maximum throughput. This is especially well recognizable with the quad-processing curve. There, almost 50 MBit/s can be processed without loss. But when the incoming traffic exceeds this threshold, the performance drops significantly. The reason for this peculiarity is the fact that the AMnode must perform a certain amount of processing even for packets that are afterwards discarded. Hence, the throughput is higher when the active node must not wast resource for discarding the surplus packets.

The measurements that have been discussed so far examined the performance for packets of 1460 byte length.

Since the packet processing involves a certain amount of additional per-packet overhead, the achievable throughput also drops, when we reduce the packet size. According to our measurements (fig. 4) a 40 MBit/s source needs to use packets of at least 700 bytes to fully exploit the performance described above. For low-data rates the packet size has a negligible impact since then the actual processing of the payload exceeds the active node's overhead by far. Furthermore, we can assume that a high data-rate source will typically use large packets, whereas small packets will be used by low data-rate streams. Thus, this limitation can be expected to not apply in practice.

Concerning the performance of our MP3-multicast reflector, we now understand why this service is capable of handling a very large number of streams without becoming overloaded. Since there no processing of the packets' payload is required, the AMnode can actually achieve the base load limit discussed above. In fact, in our lab test-bed we were always able to effortlessly support all the MP3-players that we brought into the lab. A load test where the receivers just received the stream without playing it showed that we expectedly achieve about 70 MBit/s throughput. Hence, for a typically 128 kBit/s MP3 stream we are able to support several hundreds of receivers with one 800 MHz AMnode.

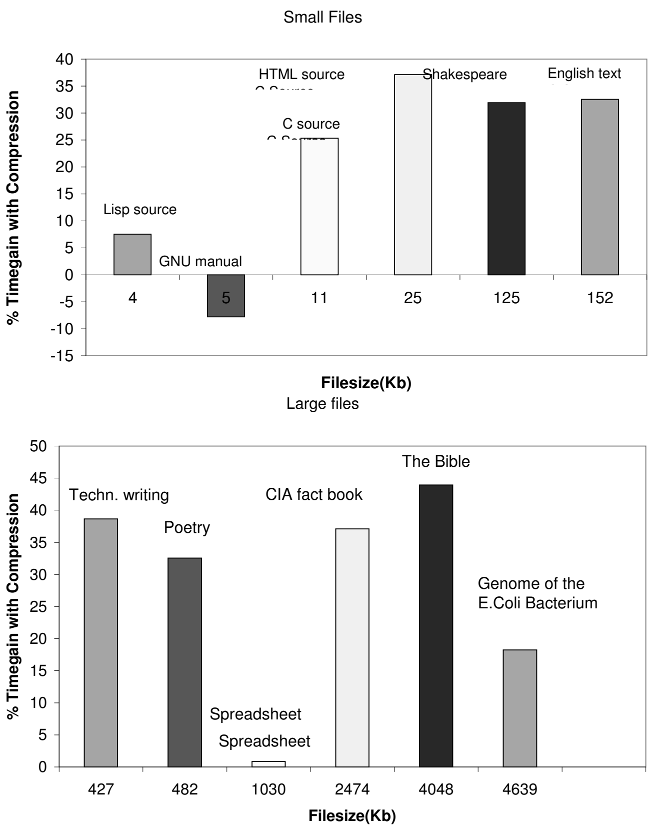

Fig. 5. Saved download time in percent (10 MBit/s wireless link)

Figure 5 shows the effect of a more sophisticated service, the HTTP compressor, loaded onto the same 800 MHz AMnode as before. For the measurement, several files were downloaded via a 10 MBit/s wireless link. The respective download time was measured as time between the first SYN packet and last FIN packet of the underlying TCP connection. Typically, this time was reduced by 25 - 30 % with a slight advantage for large files. With very small files, the HTTP compressor tends to achieve only small speed-ups. In one example, a GNU manual, the transmission was even delayed by 8 %. But larger files, e.g., a 25 kB HTML source, were transmitted at least 25 % faster. Very large files reached gains of more than 40 %.

Of course, this effect depends on the link capacity. With the 100 MBit/s Ethernet link, the transmission is always delayed. But wireless access scenarios obviously greatly benefit from the service. Especially when the link is loaded by a large number of users. In order to demonstrate that an AMnode is able to handle enough parallel connections, the following measurement was performed (fig. 6). A test machine downloaded *index.html* files from 1000 randomly selected web-servers. Each dot in the diagram represents the transmission time of a complete HTTP transfer. The slanted clouds (marked by lines) are

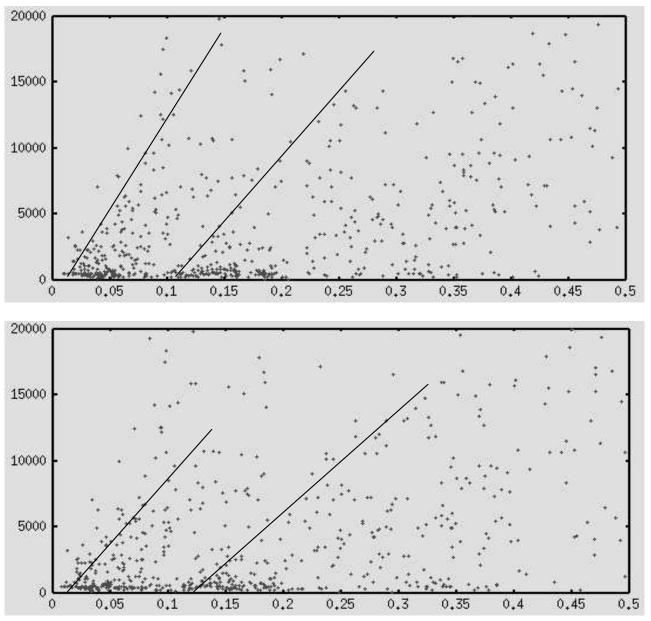

Fig. 6. Effects of the generic TCP interceptor: File size (byte) plotted over transmission time (seconds), without service (top) and with service (bottom)

caused by servers at different round-trip-time distance. The slope of the lines represent the respective transmission rate (max. 800 kBit/s).

During the measurement the active node handled up to 400 concurrent connections, most of which are of course long-lived connections to slow web-servers with far less than 800 kBit/s. Clearly, otherwise, the throughput limit of the active node would have been exceeded, anyway. Although the AMnode needs to maintain state for all connections, only the active connections cause CPU load, which is the crucial parameter for the AMnode. Hence, like with the MP3-multicast reflector, a single AMnode would greatly improve the networks performance. Furthermore, in this case, integration into a wireless access point can even avoid the installation of additional equipment and simplify the deployment of active networking technology.

The third example, the Palm video transcoding service, demonstrates that an active node has an additional advantage beyond the traditional proxy approach: It can provide specialized services for various end-devices.

Clearly, the Palm video transcoding example represents a service that exerts a high per packet processing load onto the active node. Accordingly, here, the AMnode reaches its performance limit. (A 1200 MHz machine was used for this measurement.) Currently, our service is only able to process one stream at a time. However, concerning practical feasibility of programmable networks, this is not a severe limitation since, again, we can expect such a service to be placed on an AMnode just before the low-bandwidth wireless link, e.g., as with the HTTP-compressor, an AMnode that is integrated with a Bluetooth access point. Since there the whole available bandwidth is occupied by one application, the active node needs to support only one receiver.

V. Conclusions and Outlook

This paper has described first practical results obtained from the measurement of real-world active applications. The three services that have been presented (MP3-multicast reflector, HTTP-compressor, Palm video transcoding) provide direct benefit, e.g, to an Internet Service Provider or an end-user. The measurements we conducted give evidence that these services can actually be operated with the respectively required performance.

Currently, we focus on various performance improvements that can be expected to further increase throughput and capacity of our AMnet system. Furthermore, we are planning to extend our system to especially support more low-resource devices, like the Palm PDA. Since such devices cannot themselves perform CPU or memory intensive tasks, we believe active nodes at the network's edge to be the ideal place for such services.

References

[1] Thomas Fuhrmann, Till Harbaum, Marcus Schöller, and Martina Zitterbart. AMnet 2.0: An improved architecture for programmable networks. In *Proceedings of IWAN'02*, Zürich, Switzerland, December 2002.