Deploying an Active Voice Application on a Three-Level Active Network Node Architecture

Georg Carle[1], Henning Sanneck[1], Sebastian Zander[1], and Long Le[1,2]

[1] GMD Fokus, Kaiserin-Augusta-Alle 31, 10589 Berlin, Germany
{carle, sanneck, zander, le}@fokus.gmd.de
[2] Department of Computer Science, University of North Carolina at Chapel Hill, NC 27599-3175, USA
le@cs.unc.edu

Abstract. Active networks have been recently highlighted as a key enabling technology to obtain immense flexibility in terms of network deployment, configurability, and customized packet processing. However, this flexibility is often achieved at the cost of router performance. In this paper, we present a three-level node architecture that combines flexibility and high performance of network nodes. We design and implement an active network application for real-time speech transmissions on top of this three-level platform. In our application, plug-in modules are downloaded onto certain network nodes to monitor packet loss rate of voice streams and to perform application-specific packet processing when necessary. In particular, we propose to perform loss concealment algorithms for voice data streams at active network nodes to regenerate lost packets. The regenerated speech data streams are robust enough to tolerate further packet losses along the data path so that the concealment algorithms at another downstream node or at the receiver can still take effect. We call our approach *reactive concealment for speech transmission* to distinguish it from concealment performed at the receiver and also proactive schemes like Forward Error Correction. Our approach is bandwidth-efficient and retains the applications' end-to-end semantics.

1 Introduction

We present a three-level active network node architecture to fulfil the requirements for necessary flexibility without impairing the routers' performance. The architecture is composed of three different levels. The fixed part contains components for forwarding functionality and QoS primitives. These components are optimised and static because of performance reasons. The programmable part encompasses the interfaces of the fixed part and provides abstractions of the fixed part as well as an open interface to the higher level. The active part offers a limited execution environment for lightweight active code. Our three-level architecture is very interesting for vendors of legacy routers since their proprietary interfaces can be wrapped around by a generic interface and thus integrated and migrated to an active

network environment. A prototype of our architecture has been implemented on top of Hitachi's high-speed routers GR2000.

Based on our network node architecture, we have designed an application for real-time speech transmission. Recent application-level techniques like Adaptive Packetization and Concealment (AP/C) have demonstrated that the perceived speech quality can be improved by exploiting speech stationarity [Sann98]. However as AP/C exploits the property of speech stationarity, its applicability is typically limited to isolated, i.e. non-consecutive losses. When the rate of burst losses is high, AP/C does not achieve any significant performance improvement compared to other techniques. Furthermore, it has been shown that speech quality drops significantly in the occurrence of burst losses [GS85]. We believe that this is the point where flexibility provided by active network nodes can be exploited to help applications at end systems to perform better. Our node architecture offers sufficient flexibility for programming hardware components via a generic interface to perform application-specific tasks. These tasks include monitoring of RTP streams to measure packet loss rate and enforce QoS primitives implemented in hardware to give these streams higher priority only when it is necessary. When the packet loss rate exceeds a certain threshold, a plug-in module of concealment algorithms is downloaded and performed at certain network nodes to regenerate lost packets and to inject them into voice streams. The network nodes are programmed via an open interface to perform application-specific processing in software only for the specified voice streams. Other packets passing through the network nodes are forwarded in hardware to retain high performance.

The rest of this paper is structured as follows. In section 2 we briefly review related work. Section 3 presents our three-level active node architecture that we developed within the BANG project [BANG]. We also discuss AP/C algorithm that we download to the active network nodes to perform loss concealment for voice streams as an application on top of our three-level active network platform. We then present our approach of placing active network nodes at certain locations within the network to leverage the efficiency of the receiver's concealment performance. Section 4 shows the results of a simulation study to evaluate the efficiency of our approach. In section 5 we describe the prototype implementation of our three-level active network node architecture. Finally, in section 6 we give conclusions of our work and outline potential future work areas.

2 Related Work

The concept of active networks allows users to execute codes on network nodes to meet their application-specific requirements. Another concept is to have well-defined open interfaces to network nodes and to separate their internal states from signaling and management mechanisms. This concept proposes the programmability of network nodes and thus reduces the operations that users are allowed to perform inside the network. A survey of research projects on active and programmable networks can be found in [TSS97] and [CKVV99].

Applying the concepts of active and programmable networks, application-level performance can be improved significantly thanks to the network nodes' application-specific packet processing. This is especially true for multimedia data that has a specific flow structure. Typical examples for application-specific packet processing at network nodes are media transcoding [AMZ95], media scaling [KCD00], packet filtering [BET98], or discarding [BCZ97] for video distribution on heterogeneous networks with limited bandwidth. Surprisingly, there are very few active network projects that exploit the active nodes' capability of application-specific packet processing to improve quality of Internet voice or audio transmissions. The work we are aware of is [BET98], [FMSB98], [MHMS98]. In [BET98] active nodes add an optimal amount of redundant data on a per-link basis to protect audio streams against packet loss. [FMSB98] and [MHMS98] describe the use of so called protocol boosters that allow dynamic protocol customization to heterogeneous environments within the network. A protocol booster can run on an active node and performs forward error correction to improve transmission quality over a lossy link. Similar concepts although not explicitly targeted for active networks are described in [BKGM00].

Since most packet losses on the Internet are due to congestion (except for wireless networks), we argue that it is not the most efficient method to transmit redundant data on a link that is already congested. We propose an approach where application-specific packet processing is performed at an uncongested active node to regenerate audio packets lost due to congestion at congested upstream nodes. Furthermore, the efficiency of our algorithm for lost packets' regeneration can be significantly improved when it is combined with other programmable modules. These modules include DiffServ and monitoring services that are implemented within our architectural framework. Other programmable modules are being designed and implemented.

Following the concept of programmable networks, the IEEE Project 1520 [P1520] is an effort to standardize programming interfaces for network nodes. It defines a structure of four layers with open interfaces for network nodes: physical element level (PE), virtual network device level (VNDL), network generic services level (NGSL), value-added services level (VASL). The interfaces of these four levels are called CCM (Connection and Management), L (lower), U (upper), and V (value-added) interfaces. Recently, it has been proposed to split the L-interface into a *generic abstraction* interface (L-) and a *service specific abstraction* interface (L+). The architecture presented in this paper is related to the P1520 framework proposed in [LDVS99]. The P1520 interfaces described in [RBWYK00], [BVKV00] map quite well with our architecture. Currently further development of our architecture is being carried out within the FAIN project [BANG], [FAIN], [GPLD00].

3 A Three-Level Active Network Node Architecture

Despite immense flexibility gain, implementing low-level forwarding and QoS primitives within active components do not seem to be realistic in the foreseeable future due to performance constraints. This consideration leads to the introduction of our three levels. The three-level architecture achieves the necessary flexibility without

impairing the router's performance. The key design of our architecture is to de-couple the control software from the forwarding functionality implemented in hardware. The fixed level of our architecture contains static and optimized forwarding components with QoS primitives that cannot be made programmable due to performance reasons. In our node architecture, data packets that make up for the majority of packets flowing through a network node are processed directly by hardware in the fixed level. The programmable level exploits the primitives and high performance of the fixed part to provide end-to-end services with an open interface. Control packets without application-specific requirements are serviced by generic network mechanisms in the programmable level. In order to provide a good perceived quality, multimedia applications typically require that a certain end-to-end quality of service is guaranteed. The programmable part can fulfil the requirements of quality of service by enforcing Differentiated Services primitives [BBC98] in a co-ordinated way between network nodes. Monitoring of streams' packet loss rate is another important service of the programmable level.

Exploiting the monitoring service, the application level can download and execute plug-in modules only on necessary network nodes. In our architecture, services of the programmable level are implemented on top of the fixed part. The node-local interface to the fixed part is implemented by establishing an automated telnet connection to Hitachi's gigabit routers to perform the necessary router configuration. By doing this, we separate the data path and the control path to avoid performance loss. The programmable level is in turn controlled by the active level via an open interface. The active level offers a limited execution environment for lightweight active codes. Lightweight active codes are usually application-specific algorithms. Lightweight active codes use the module interfaces of the programmable part to access the functionality of the fixed part implemented in hardware. Lightweight active code typically contains function calls or simple scripts to the module interfaces of the programmable part with specific parameters. Other active codes contain simple instructions for downloading programmable modules onto network nodes. After these modules are downloaded, subsequent active codes carry along only parameters necessary to configure these modules.

3.1 Mapping of P1520 Interfaces to the Three-Level Active Node Architecture

A network node in the three-level active node architecture consists of a hardware-dependent and a hardware-independent part. The main idea here is to develop a framework for programmable routers via a generic interface. However this generic framework exploits hardware specific features (such as the QoS primitives of Hitachi GR2000 gigabit router) to achieve high performance. Thus, the network node's programmability is generic while active codes can still exploit network node's specific features.

Our architecture stands in close relation with the proposed P1520 framework [LDVS99]. Fig. 1 shows a view on the three-level active node architecture with relation to the P1520 interface specifications. In our architecture, an automated telnet connection between the GR2000 router and a PC router controller is the interface

between the fixed and the programmable level. The GR2000 on one side of the telnet connection forms the active node's hardware-dependent part. The PC router controller is a Linux box running the active node's software on the other side. It furthermore offers limited execution environments for plug-in modules that could burden the GR2000 resources otherwise. The telnet interface is roughly equivalent to the *CCM-interface* of P1520. On the hardware-dependent side, it is accessed by a telnet C library that enables the configuration of the GR2000 hardware's QoS and filtering primitives. This library is in turn wrapped by a Java library to leverage active code from hardware and operating systems' dependencies. The GR2000 Java interface is used to configure low-level router primitives. This interface could be identified to be at the L- level because it abstracts from the device-specific GR2000 interface, yet it does not completely fit into the P1520 framework. On top of the L- interface higher level QoS modules are implemented. These modules first abstract from specific devices and offer an interface to software architectures at the L+ level. They have purely local as well as service-specific semantics.

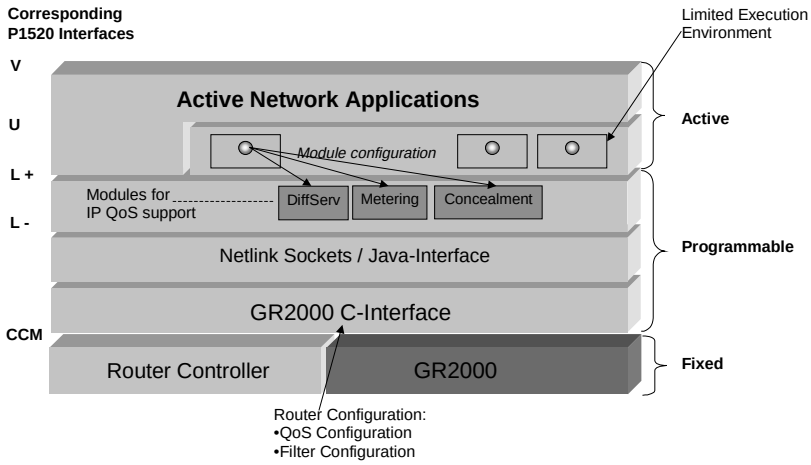

Fig. 1. Three-level active node architecture

Active network applications trigger the installation of active components using the end-to-end *V interface*. The installation consists of transporting the active components to the remote active nodes using a mobile agent. The transport mechanism is implemented by a Java mobile agent platform that was modified for this special purpose. Application designers are shielded from implementation details of the mobile agent platform and only have access to a network device via an abstract interface. The virtual network device level realizes the code transport mechanisms to the *local node* and limits the access to *local resources* of the GR2000 routers as well as the PC router controller. Thus, the *L interface* can be used by application designers to access resources of the GR2000 routers and the PC router controller without detailed knowledge about code transport mechanisms or router configurations. Under the control of a host manager the active components are installed and executed. At the U level network-wide services can be accessed by active code. Finally, interfaces at

the V level allow applications to combine services of lower levels. For example, an application can configure and access the meter modules of programmable nodes to estimate the packet loss rate along the path of a multimedia stream. Having this information, it can either enforce the routers' QoS capability at the bottleneck link or download and perform a concealment algorithm at another programmable network node downstream of the bottleneck link.

3.2 DiffServ Module

To achieve interoperability with standard QoS control mechanisms, programmable routers' interfaces allow resource control by traditional QoS control as well as by components dynamically deployed by active networks. The three-level active node architecture is designed to offer such an open QoS control interface. Fig. 2 shows the investigated architecture. It is based on our three-level active node architecture, shown on the right side of the figure, but extends the proprietary GR2000 router interface with a standardized DiffServ interface for QoS control specified by the L+ level.

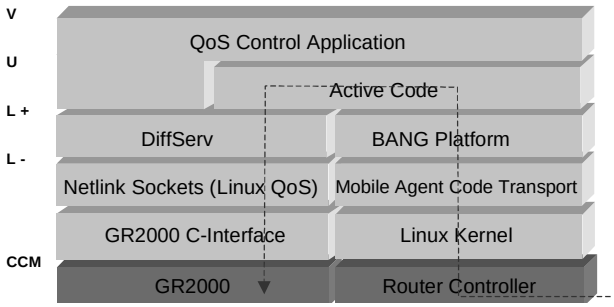

Fig. 2. Three-level active node architecture and DiffServ interface

Instead of accessing the GR2000 via a proprietary and router vendor specific interface, active components control the router via a DiffServ API. The DiffServ API also enables non-active QoS control applications to configure a GR2000 router. On one side, this API allows an easy realization of the proposed architecture. On the other side, it allows future porting of the architecture to other routers. Comparing our architecture with the P1520 interface proposal, the L interface is divided into two sub-layers: L+ providing a standardized QoS control interface and L- providing a mapping from the DiffServ layer to the router specific API.

This DiffServ interface definition was implemented using the C programming language. The DiffServ C-interface allows programs to control the router by setting and deleting the DiffServ structures defined by the L+ interface. Because active code is implemented in Java within our three-level active node architecture, it is necessary to wrap the developed C-interface by a Java interface providing similar functionality,

i.e. implement the L+ IDL definition in Java. Thus, legacy programs can use the C-interface whereas active programs may use the Java-interface.

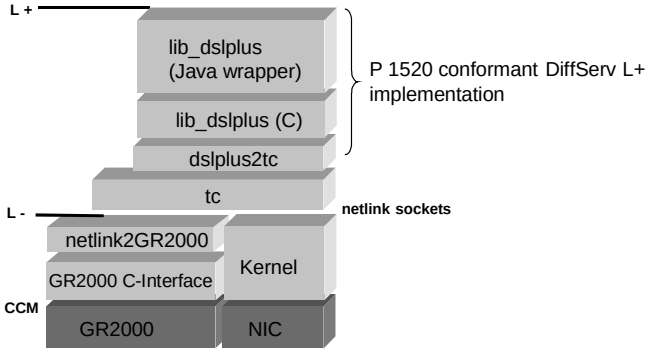

Fig. 3. DiffServ module structure

Fig. 3 shows the internal layering of the DiffServ module. Note that besides access via the L+ interface, it is also possible to use other lower interface levels. In addition to the P1520 L- interface also intermediate levels exist, namely the GR2000 C (or Java) interface and the 'tc' tool which is a well-known traffic control tool to configure the Linux QoS interface. The 'tc' tool is implemented beyond the Linux netlink interface which can be considered equivalent to the CCM interface. The DiffServ implementation presented in this paper realizes a standardized interface for QoS control on Hitachi's GR2000 router. The DiffServ interface is implemented in Java and C for providing access to traditional as well as active management components. Section 5 will demonstrate how active networks can be used to provide QoS to multimedia applications by autonomous components deployed in case of network congestion.

3.3 Active Meter Module

In traditional metering systems, data is gathered by distributed meters in the network. The metering data is collected by readers and transferred to some management entity on an end system. The RTFM architecture [RFC2063] is an example for such a traditional system. In an active network we have to deal with mainly two additional issues. First, active networks enable the rapid deployment of new protocols and services within the network. This means that even end users might be able to introduce new protocols and services. Active metering must deal with these new protocols and services. Therefore an active meter must be flexible and extensible. In the extreme case that the user is able to deploy a new protocol, it must be possible to meter this protocol for testing, accounting and management purposes.

The second issue arises from the possibility of having active code running on the active nodes. If this active code is performing active QoS enforcement or improvement (Protocol Enhancing Proxies (PEPs) [BKGM00] or Protocol Boosters [FMSB98]), it needs access to local metering data. One example for a PEP is the Adaptive Packetization and Loss Concealment (see section 3.4) method which

improves Internet voice transmission in the case of packet loss. A PEP needs access to metering data for two reasons. Firstly, the PEP must decide when to become active. In the case of no loss or very high loss rates, activation of AP/C does not improve the voice quality but instead introduces additional delay and wastes CPU and memory resources on the active node. Secondly, the PEP operation might be parameterized through metrics measured. In the case of improving the quality on links with packet loss, Forward Error Correction (FEC) can be used. To avoid wasting resources, the amount of redundancy generated must depend on the current measured loss rate.

General requirements for a meter system are speed and efficiency. This means that the effort for metering should be low compared to the effort of packet forwarding. Active metering should have a minimal impact on the performance of the active node. Therefore we propose *native code modules* as the most promising tradeoff between flexibility and performance. The selection of programmable modules for a certain task is specified in a rule set language. This approach also supports a heterogeneous infrastructure assuming that we can provide native modules with the same function for each device. In that sense the meter is active because it can be dynamically extended and enhanced with new modules providing the needed functionality.

The design of the active meter maps to the P1520 architecture described in [LDVS99]. For the L- Layer it was decided to use Linux NetFilter [Russ00]. Fig. 4 depicts the architecture with relation to the P1520 layers. The basis of the architecture is the Linux kernel. The CCM interface separates the NetFilter code from the rest of the kernel code. It is not a clear interface but rather consists of a number of function calls. On top of the CCM interface is the NetFilter code. NetFilter is controlled via a userspace tool called *iptables*. This tool represents the L- interface to the NetFilter classification functionality. On top of the L- interface, the meter core consists mainly of a flow table and a ruleset manager. The control interface allows the management of the ruleset within the meter. Rules can be added, modified or deleted. For defining rulesets a simple ruleset language has been defined. The data interface allows access to the meter data. Based on the flow specification the metered data like byte and packet counter or the current packet loss rate can be retrieved. The control interface and the data interface together form the L+ meter interface. The active meter has been implemented in C.

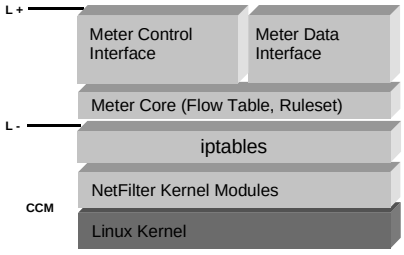

Fig. 4. Active meter module structure

The loss rate measurement functionality of the active meter can be exploited by modules performing active QoS enhancements. In this paper we propose the Adaptive Packetization and Concealment (AP/C) algorithm for improving voice transmission over links with moderate loss rates. Since the GR2000 is not capable of metering on

flow level granularity, the active meter resides on the router controller. A small set of rules is enforced in the GR2000's hardware. Packets that match the rules are passed to the router controller for metering purposes. Other packets are forwarded by the GR2000 as usual. The router controller meters the packets passed by the GR2000 and routes them back to the GR2000 via another network interface to avoid packet looping. Active codes can access the meter module to set rules and retrieve metering data via an open interface. For example a mobile agent can be used to set up or change the meter rule set or to read data and transport it to another system.

3.4 Loss Concealment Module

AP/C (Adaptive Packetization / Concealment) exploits the speech properties to influence the packet size at the sender and to conceal the packet loss at the receiver. The novelty of AP/C is that it takes the phase of speech signals into account when the data is packetized at the sender. AP/C assumes that most packet losses are isolated. In AP/C, the receiver conceals the loss of a packet by filling the gap of the lost packet with data samples from its adjacent packets. Regeneration of lost packets with sender-supported pre-processing works reasonably well for voiced sounds thanks to their quasi-stationary property. Regeneration of lost packets works less well for unvoiced sounds due to their random nature. However, this is not necessarily critical because unvoiced sounds are less important to the perceptual quality than voiced signals. Since the phase of the speech signal is taken into account when audio data is packetized, less discontinuities than for conventional concealment algorithms are present in the reconstructed signal.

Since AP/C assumes that most packet losses are isolated, it does not obtain any significant performance improvement compared to other techniques when the rate of burst losses is high. We believe that this is the point where the active nodes' capability of application-specific packet processing can be exploited to help applications at end systems perform better. Since the burst loss rate of a data flow at a network node is lower than at the receiver, the AP/C concealment algorithm works more efficiently and more lost packets can be reconstructed when concealment is performed within the network rather than just at end systems. We thus propose to download and perform the AP/C concealment algorithm at certain active nodes where the number of burst losses of a voice data stream is sufficiently low to regenerate the lost packets. The regenerated audio stream is robust enough to tolerate further packet losses so that the AP/C concealment algorithm can still take effect at another downstream active node or at the receiver.

The idea of the active network application is demonstrated in Fig. 5. The AP/C sender algorithm is performed to packetize audio data taking the phase of speech signals into account. Along the data path, packet 2 and 4 are lost. Exploiting the sender's pre-processing, the AP/C concealment algorithm is applied at an active node within the network to reconstruct these lost packets. Downstream of the active node, another packet is lost (packet 3) which is easily reconstructed at the receiver. In this scenario, active concealment reconstructs six lost "chunks" (a chunk is a logical unit of speech identified by the AP/C sender algorithm; in Fig. 5 they are designated by

c_{21}, c_{22}, c_{31}, c_{32}, c_{41}, and c_{42}) and clearly outperforms the receiver concealment [Sann98] which can only reconstruct at most two chunks (c_{21} and c_{42}) due to the burst loss accumulated along the end-to-end data path.

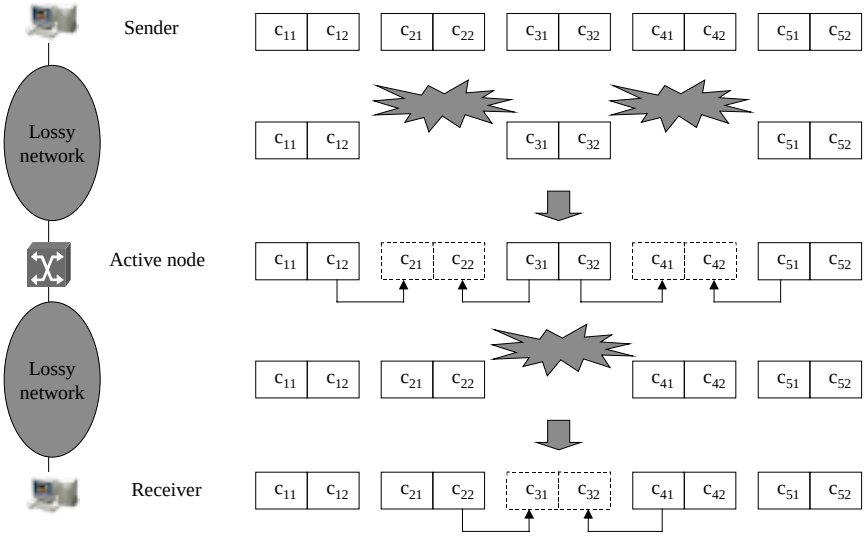

Fig. 5. Active concealment

Thanks to flexibility offered by active nodes, the plug-in module for concealment can operate in a spectrum from the observer mode (only regenerate lost packets) to the proxy mode (buffer or recover multiple packets and then forward them). Observer mode consumes less CPU and memory resources of an active node but proxy mode is more powerful.

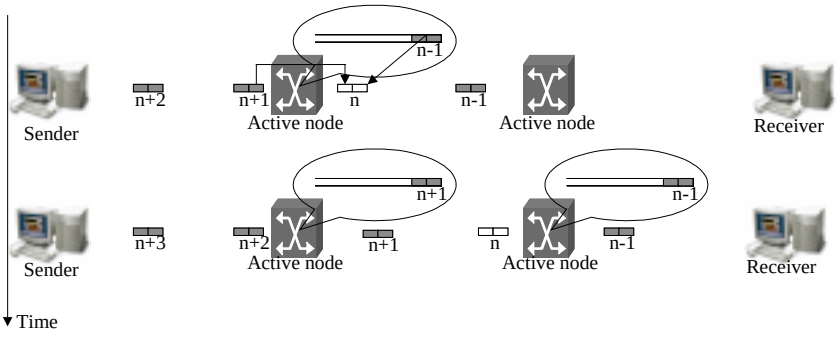

Fig. 6. Active nodes in observer mode

In the observer mode, an active node buffers only the packet with the highest RTP sequence number of a flow. When the gap in the RTP sequence number between a new packet and the currently buffered packet is larger than one, the active node assumes that a burst loss has occurred upstream. It throws away the old packet and

buffers the new one. The underlying assumption here is that out-of-order packets are rare. An active node delays the current packet until the lost packet has been reconstructed with AP/C to avoid reordering. The advantage is that downstream active nodes can avoid duplicate concealment. Duplicate concealment would happen if a single packet is lost and every active node along the path attempts to conceal the lost packet when it sees the previous and the next packet. Fig. 6 shows a scenario with two active network nodes in the observer mode. Since the lost packet is reconstructed from its previous and next packet, it suffers an additional delay. This additional delay is exactly the delay between two subsequent packets. The additional delay incurred by packet regeneration is demonstrated in Fig. 7. Let d be this delay, n be the number of times a packet can be regenerated, and D be the maximum playout budget[1] of the receiver. We have the constraint $n*d \leq D$. Thus, the number of active nodes along the path of a voice flow should be smaller than $\lfloor D/d \rfloor$. Otherwise, packets regenerated more than $\lfloor D/d \rfloor$ times are discarded by the receiver because they arrive later than their playout time. Consider an interactive voice application of two users between Chicago and Paris. Interactive voice application requires that one-way delay be smaller than 250 ms. The distance between Chicago and Paris is 4142 miles which translates into an approximates propagation delay of 4142 * 1600 / 300000000 ≈ 22 ms. Let the queueing delay be 35 ms, the maximum playout delay is D = 250 – 22 – 35 = 193 ms. Depending on speaker's voice, an average AP/C packet size ranges from 80 to 160 samples [Sann98]. In this example, we choose a packet size of 120 samples and obtain a packetization delay of 120 / 8000 = 0.015 s = 15 ms. Thus, the maximum number of active nodes on a voice path between Chicago and Paris is $\lfloor 193/15 \rfloor = 12$.

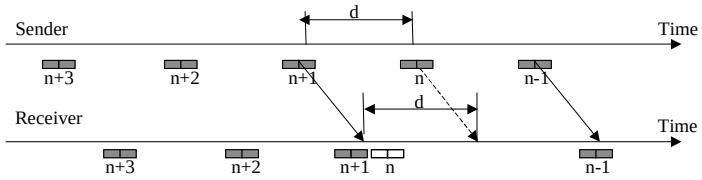

Fig. 7. Additional delay incurred by packet regeneration

In the proxy mode, an active node buffers more than one packet of a voice flow. An active node can determine how long packets are buffered for the concealment operation. The longer packets are buffered at an active node, the more memory is consumed but the higher the chance of a successful reconstruction is. This trade-off is similar to that of a receiver's playout buffer. When a loss gap of one packet is detected, the lost packet is regenerated as described in the observer mode. Upon detecting a burst loss larger than one packet, an active node requests its upstream node to retransmit the lost packets. Fig. 8 illustrates a scenario where packets n and n+1 are lost and retransmitted by an active node operating in proxy mode. Since the proxy mode can cope with burst loss, it outperforms the observer mode. An active node limits the number of voice packets kept in its buffer and replaces the old packets by the new ones. An active node can also periodically send its upstream neighbor an

[1] Playout budget is the maximum amount of time a packet can be kept in the receiver's playout buffer.

active packet acknowledging voice packets up to a certain RTP sequence number. An acknowledgement does not mean that an active node has received and forwarded all packets up to the specified RTP sequence number. It rather means that the active node does not need those packets any more. This acknowledgement mechanism helps limit the number of buffered packets at an active node. Let k be the maximum recoverable burst loss and rtt_i be the roundtrip time between the *i*th active node and its upstream neighbor (in the proxy mode, the sender also participates in the ARQ process and is considered the *0*th active node). We have the constraint

$$n*(k+1)*d + rtt_1 + rtt_2 + \ldots + rtt_n = n*(k+1)*d + rtt \leq D$$

where rtt is the roundtrip time between the sender and the receiver. This inequality constraint presents a trade-off between the number of active node on the path of a voice flow and the maximum burst loss we wish to recover. Similar to the example of the observer mode, we have $n*(k+1)*15 + (22+35)*2 \leq 193$. Thus, $n*(k+1) \leq \lfloor 79/15 \rfloor = 5$. If we have only one active node operating in proxy mode on a voice path between Chicago and Paris, we can recover a burst loss up to four packets.

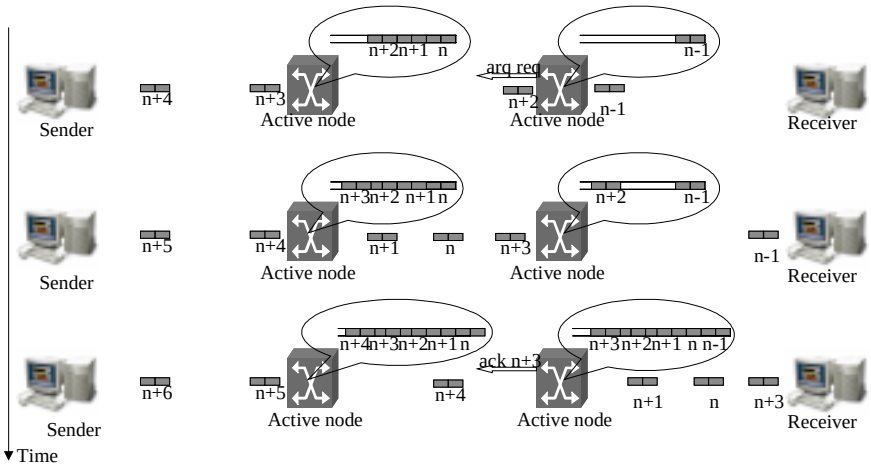

Fig. 8. Active nodes in proxy mode

Our approach is similar to Robust Multicast Audio (RMA) proposed by Banchs et. al. in [BET98] but it acts in a *reactive* way upon detection of packet loss in audio data streams. On the contrary to RMA transmitting redundant data on a per-link basis to protect audio streams against packet loss in a proactive way, our approach simply regenerates and injects the lost packets into audio streams and thus is more bandwidth-efficient. Another advantage of our approach is that it does not break the applications' end-to-end semantics and does not have any further demand on the number and location of active nodes performing the concealment algorithm[2]. RMA, however, requires active nodes to be located at both ends of a link or a network to

[2] Clearly, the number and location of active network nodes influence the performance improvement. However, the applications' functionality is not affected under any circumstances.

perform the FEC encode and decode operation. The concept of protocol boosters [FMSB98], [MHMS98] is another similar approach to ours. However, in the case of FEC which is presented in [MHMS98] this approach also requires at least two instances of the same booster type within the network in order to perform the encode and decode operation in a proactive way. On the contrary, our scheme is reactive and only acts when necessary.

4 Simulation Study for Active Concealment

In our simulation, we assume that there is only one active node in the path from the sender to the receiver where intra-network regeneration of lost packets can be performed. The logical network topology for our simulation is shown in Fig. 9 where a lossy network can consist of multiple physical networks comprising several network hops. We use the Bernoulli model to simulate the individual loss characteristics of the networks. Objective quality measurements such as [ITU98] and [YKY99] are used to evaluate the speech quality. These measurements employ mathematical models of the human auditory system to estimate the perceptual distance between an original and a distorted signal[3]. Thus, they yield results that correlate well and have a linear relationship with the results of subjective tests. We apply the Enhanced Modified Bark Spectral Distortion (EMBSD) method [YKY99] to estimate the perceptual distortion between the original and the reconstructed signal. The higher the perceptual distortion is, the worse the obtained speech signal at the receiver is. The MNB scheme [ITU98], though showing high correlation with subjective testing, is not used because it does not take into account speech segments with energy lower than certain thresholds when speech distortion is estimated. In the MNB scheme, the replacement of a lost speech segment by a silent segment does not lead to a degradation of quality, because this segment is not taken into account when the perceptual distortion is computed.

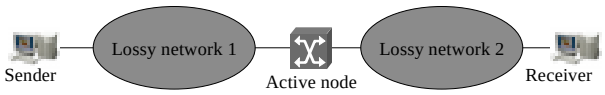

Fig. 9. Simulation topology

The structure of this section is organized as follows. In the first simulation step, we use the same parameter sets for the lossy networks. We then compare the speech quality obtained by the active loss concealment with two reference schemes. In the second simulation step, we vary the parameter sets of the lossy networks and measure the efficiency of the active loss concealment. The parameter sets are chosen in such a way that the packet loss rate observed at the receiver is constant. This simulation step is performed to determine to optimal location of the active node where the plug-in module for the active concealment algorithm can be downloaded and performed.

[3] We use a speech sample that consists of different male and female voices and has a length of 25 s.

4.1 Performance Comparison to Reference Schemes

In this simulation step, we compare the speech quality obtained by active loss concealment with two reference schemes. In the first reference scheme, the sender transmits voice packets with constant size and the receiver simply replaces data of a lost packet by a silent segment with the same length. Each packet in this scheme contains 125 speech samples, resulting in the same total number of packets as the second reference scheme and the active loss concealment scheme. The second reference scheme is the AP/C scheme applied only at end systems. Packets are sent through two lossy network clouds and are dropped with the same packet drop probability. The parameters used in this simulation step and the resulting packet loss rate are shown in Table 1.

Packet drop probability	0.03	0.06	0.09	0.12
Packet loss rate	0.0592	0.1164	0.1720	0.2257

Table 1. Parameters and packet loss rate used in simulation for performance comparison

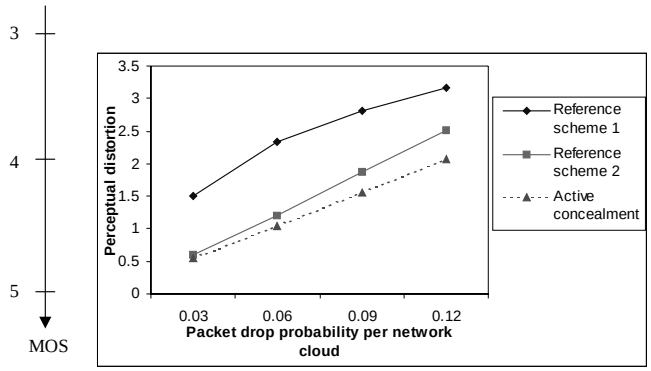

Fig. 10. Performance comparison to reference schemes (simulation step 1)

Fig. 10 shows the results of this simulation step, plotting the perceptual distortion measured by EMBSD versus the network clouds' packet drop probability. The MOS (Mean Opinion Score) axis helps the reader to interpret the results in term of subjective quality measurement. A MOS value of 5 indicates excellent speech quality while a MOS value of 1 stands for an unacceptable quality. The results demonstrate that the higher the packet drop probability is, the higher the perceptual distortion of the schemes and thus the worse the speech quality is. AP/C performs better than reference scheme 1 that replaces lost packets by silent segments, and the active loss concealment obtains the best speech quality. When the network clouds' packet drop probability is low, the active loss concealment does not gain any significant improvement compared to the AP/C scheme. This is because AP/C performs sufficiently well when the network loss rate is low and the number of burst losses is negligible. However, when the packet drop probability rises and the burst loss rate is no longer negligible, the perceptual distortion obtained with AP/C increases significantly and the active loss concealment achieves obvious improvement compared to AP/C.

4.2 Optimal Active Network Node Location

In this simulation step, we vary the parameters of the lossy network clouds to determine the optimal location of the active network node. This simulation step is intended to help answering the following question: „Given that there are the same loss characteristics along the data path, where is the most effective location to download and perform the active concealment algorithm?“.

The packet loss rate p of a data path consisting of two network clouds with packet drop probability p_1 and p_2 is given by $p = 1-(1-p_1)\cdot(1-p_2)$.

The result of this simulation step is presented in Fig. 11 using EMBSD to compute the perceptual distortion of the obtained speech signal at the receiver. It shows that the optimal location to download and perform the active concealment algorithm is where the packet loss rate from the sender to that location is equal to the packet loss rate from there to the receiver ($p_1 = p_2$). If on one hand the packet loss rate from the sender to the location of the active node is too high ($p_1 >> p_2$), the active concealment algorithm cannot exploit its advantage in terms of the location as compared to concealment just at the receiver. On the other hand, if the packet loss rate from the active node to the receiver is too high ($p_1 << p_2$), the concealment algorithm at the active node is idle most of the time, because the majority of losses happen at subsequent network nodes. This effect is increasingly important when the packet loss rate (and thus the packet drop probability) increases, leading to a higher number of burst losses which causes the “conventional” concealment algorithm to fail.

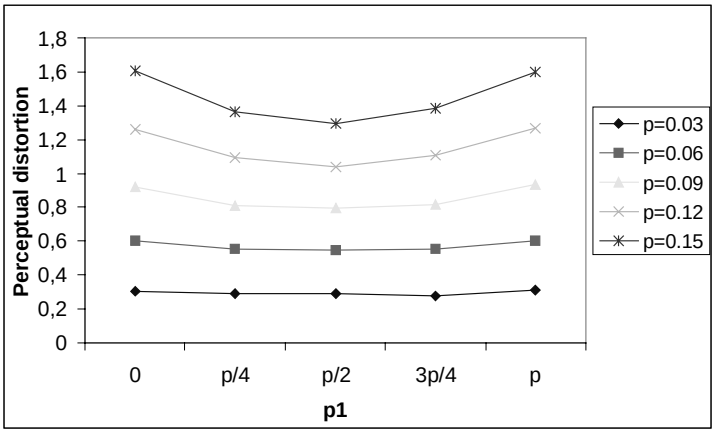

Fig. 11. Optimal active network node location (simulation step 2)

5 Prototype Implementation

We have set up a testbed for the purposes of demonstration, experimenting, and implementing applications that can exploit the advantages of active networks

technologies. The heart of the testbed are three GR2000 routers provided by Hitachi. These three high-speed routers satisfactorily fulfil the requirements for large bandwidth of novel applications and also provide primitives for QoS and filtering. Each of these routers is connected to a PC router controller that controls the GR2000 via the interface described in section 3. The software of our architecture runs on the router controller. Metering is also performed on the router controller because the current version of the GR2000 has no flow based metering capabilities. Our testbed is shown in Fig. 12.

In the following we describe a scenario of DiffServ service creation, deployment, enforcement and measurement which we have realized within the testbed. The scenario comprises the following features:

- application of active concealment for voice flows when the packet loss rate is low (a threshold of 10% is used in our current testbed)
- creation and deployment of state to DiffServ core routers realizing a Per-Domain Behaviour (PDB)
- creation and deployment of per-flow state in DiffServ edge routers to map a flow to a DiffServ Code Point (DSCP)
- flow remarking and enforcement of the DiffServ Per-Hop Behavior (PHB) using the automated router configuration and measurement (active meter)

When foreground traffic (i.e. in this case it is voice traffic) is transmitted, the measurement system at the core router detects the foreground flow. However it is not able to decode the upper layer protocol (RTP). Therefore the corresponding meter module is requested and uploaded from the network management system. Then measurement data on the flow is collected. When a Per-Domain Behaviour (PDB) is created at the management station, it can be deployed by having mobile agents travel to core routers to perform the necessary configuration for the respective PHBs. When the packet loss rate is lower than 10%, no reservation of bandwidth is necessary and active concealment is applied to regenerate lost packets as described in section 3.4. When congestion is about to occur, it can be detected since the packet loss rate and routers' queue length (among other values) are measured at the routers and periodically conveyed to the network management system. Collected information at the management system can help to detect the location of congestion early and take the most appropriate measures. Upon detection of congestion, mobile agents can carry a filter to edge routers to remark the foreground traffic flow to a certain DSCP. According to the deployed PHB, the core router now gives preference to packets carrying this DSCP. Another possibility is to have mobile agents only configure the routers at the congestion location to serve foreground traffic with higher priority. Thus, partial QoS enforcement can be combined with active concealment. The voice quality now remains high independent of the congestion situation in the network. Note that it is possible to apply the filter (as well as the core routers' configuration) only temporarily in our architecture. This allows us to deploy an agent that can configure the router and become inactive for specified time interval. After that the agent becomes active again, deletes the filter, and terminates or moves to another network node.

The above scenario of automatic QoS deployment demonstrates the importance of the decentralised control of routers. This is achieved by having mobile agents move

from hop to hop to configure the routers. The decentralised control of core routers allows for simple autonomous PDB deployment. The decentralised control of edge routers allows for simple autonomous enforcement of temporary Service Level Specifications (SLSs). Due to the agent-based approach, the system has a high fault tolerance in case links from the central network management system to nodes are temporarily down or congested. Furthermore the scenario shows how our architecture's open interfaces can be used to set up a QoS configuration in an active network. Our DiffServ scenario demonstrates the main benefits from the active network technology: the autonomous time-dependent management of QoS by active components.

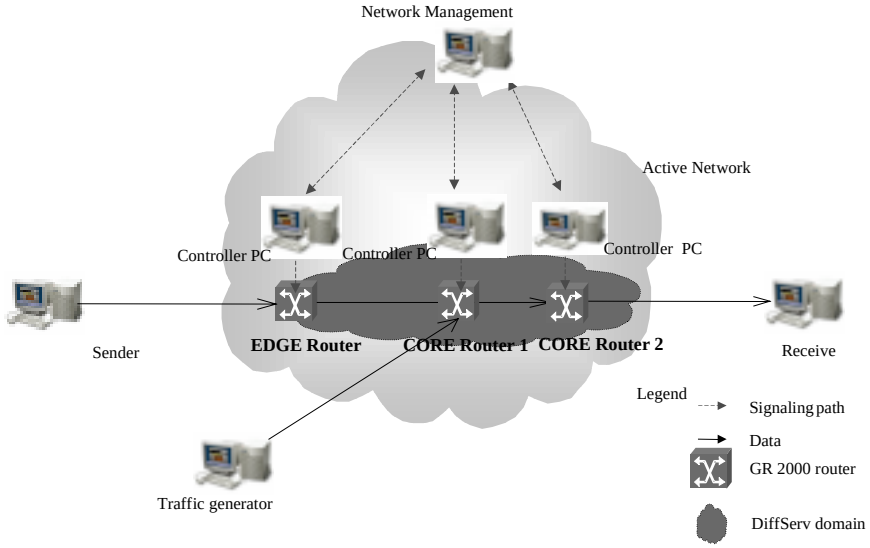

Fig. 12. Testbed configuration

6 Conclusion and Future Work

In this paper we presented a novel three-level active node architecture which consists of a fixed, a programmable and an active part. Our active node architecture achieves both flexibility and high performance that are the core requirements for a dynamic network infrastructure. Our prototype implementation bases on Hitachi high-speed routers GR2000. However, the flexibility of our network node architecture enables easy plug-in of other high-speed routers. The programmable part currently consists of a Diffserv and a meter module. The QoS module for DiffServ as part of our three-level active node architecture allows for interoperability with standard QoS control mechanisms, which is essential for the acceptance of the new active network technology. The standard interfaces allow the control of networks by traditional QoS control as well as by components dynamically deployed by active networks. We have

specified and implemented a Differentiated Services module that allows to abstract from a particular network device and to use DiffServ semantics to program the device. The meter module uses native code modules that can be dynamically loaded to extend the meter functionality. This allows for high flexibility and extensibility as well as performance. A generic ruleset language is used for specifying meter rules and corresponding modules independently from the architecture of the network elements. Mobile agents can be used for the transport of rulesets to the meter devices or for reading meter data and transport them to other systems. Other plug-in modules are being designed and implemented.

Another plug-in module for reactive concealment of voice streams is currently being implemented. With this module we design a new active network application for voice over IP that exploits the flexibility of active networks to perform application-specific packet processing. Simulation results have demonstrated that significant speech quality improvements can be achieved compared to pure end-to-end application-level algorithms. An unoptimized software implementation of the active loss concealment reconstructs a lost packet with an average execution time overhead of 220 μs on a PC with a Pentium III 500 MHz CPU and 128 Mbytes RAM. Since the active node only performs packet regeneration for a small portion of packets of voice streams, the average consumption of node resources is reasonably low. The more complex encoding can be done at the sending end system that usually has sufficient processing power because the bandwidth overhead due to encoding is very low. Together with active metering this approach allows for automatic QoS improvement for voice transmission in an active network environment.

Additional future work includes the investigation of active network applications where a number of active nodes can be placed along the data path to download and perform the active loss concealment algorithm. Besides, it is very interesting to attempt to answer the question how well and how many times active loss concealment can be performed in a recursive way. Furthermore, since both application-level Forward Error Correction and application-specific packet processing incur additional consumption of network resources, we plan to compare these two approaches. The result of this comparison might enable an optimal combination of the two approaches to obtain further improvement of speech quality. A further step is to implement plug-in modules for audio compression and Forward Error Correction. An integration of these modules in the existing programmable level allows significant flexibility and efficient coordination with other implemented modules.

References

[AMZ95] E. Amer, S. McCanne, and H. Zhang. An Application Level Video Gateway. Proceedings of ACM Multimedia 95, San Francisco, CA, November 1995.

[BANG] Hitachi, GMD Fokus. Broadband Active Network Generation (BANG), http://www.fokus.gmd.de/research/cc/glone/projects/bang/, September 2000.

[BBC98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. IETF RFC 2475, December 1998.

[BET98] A. Banchs, W. Effelsberg, C. Tschudin, and V. Turau. Multicasting Multimedia Streams with Active Networks. In Proceedings IEEE Local Computer Network Conference LCN 98, Boston, MA, Oct 11-14, 1998, pp. 150-159.

[BCZ97] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura. An Architecture for Active Networking. High Performance Networking (HPN 97), White Plains, NY, April 1997.

[BKGM00] J. Border, M. Kojo, Jim Griner, G. Montenegro. Performance Enhancing Proxies, IETF Internet Draft <draft-ietf-pilc-pep-02.txt>, March 2000.

[BVKV00] J. Bitwas, J. Vicente, M. Kounavis, D. Villela, M. Lerner, S. Yoshizawa and S. Denazis. Proposal for IP L-Interface Architecture, Draft IEEE P1520, January 2000.

[CKVV99] A. T. Campbell, M. E. Kounavis, J. Vicente, D. Villela, K. Miki, and H. De Meer. A Survey of Programmable Networks, ACM SIGCOMM Computer Communication Review, Vol. 29 No. 2 pg. 7-24, April 1999.

[FAIN] Hitachi, GMD, UCL, ETH Zürich et. al., FAIN - Future Active IP Networks, IST project, http://www.ist-fain.org/, work in progress.

[FMSB98] D. C. Feldmeier, A. J. McAuley, J.M. Smith, D. S. Bakin, W. S. Marcus, T. M. Raleigh. Protocol Boosters, IEEE JSAC, April 1998.

[GPLD00] A. Galis, B. Plattner, E. Moeller, J. Laarhuis, S. Denazis, C. Klein, J. Serrat, G. T. Karetsos, C. Todd. A Flexible IP Active Networks Architecture, IWAN 2000 Conference, October 2000.

[GS85] J. Gruber and L. Strawczynski. Subjective Effects of Variable Delay and Speech Clipping in Dynamically Managed Voice Systems. IEEE Transactions on Communications, Vol. COM-33(8), August 1985.

[ITU98] Objective Quality Measurement of Telephone-Band (300-3400 Hz) Speech Codecs. ITU-T Recommendation P.861, February 1998.

[KCD00] R. Keller, S. Choi, D. Decasper, M. Dasen , G. Fankhauser, B. Plattner. An Active Router Architecture for Multicast Video Distribution. Proceedings IEEE Infocom 2000, Tel Aviv, Israel, March 2000.

[LDVS99] P. Lin, S. Denazis, J. Vicente, M. Suzuki, J. P. Redlich, F. Cuervo, J. Biswas, W. Weiguo, K. Miki, J. Gutierrez. Programming Interfaces for IP Routers and Switches, an Architectural Framework Document, IEEE P1520/TS/IP-003, June 1999.

[LSCH00] L. Le, H. Sanneck, G. Carle, and T. Hoshi. Active Concealment for Internet Speech Transmission, IWAN 2000 Conference, October 2000.

[MHMS98] W. S. Marcus, I. Hadzic, A. J. McAuley, and J. M. Smith. Protocol Boosters: Applying Programmability to Network Infrastructure, IEEE Communications Magazine, vol. 36, no. 10, pp. 79--83, Oct. 1998.

[P1520] Proposed IEEE Standard for Application Programming Interfaces for Networks, http://www.ieee-pin.org

[RBWYK00] M. Raguparan, J. Biswas, W. Weiguo, S. Yoshizawa, A. Karlcut, "L+ Interface for Routers that Supports Differentiated Services", IEEE P1520/TS/IP-012, January 2000.

[RFC2063] N. Brownlee, C. Mills, G. Ruth: „Traffic Flow Measurement: Architecture", RFC2063, January 1997.

[Russ00] P. Russel: " Linux 2.4 Packet Filtering HOWTO", http://netfilter.samba.org/unreliable-guides/packet-filtering-HOWTO.html, May 2000.

[Sann98] H. Sanneck. Adaptive Loss Concealment for Internet Telephony Applications. Proceedings INET 98, Geneva/Switzerland, July 1998.

[TSS97] D. Tennenhouse, J. Smith, D. Sincoskie, D. Wetherall, G. Minden. A Survey of Active Network Research. IEEE Communications, January 1997.

[YKY99] W. Yang, K. R. Krishnamachari, and R. Yantorno. Improvement of the MBSD by Scaling Noise Masking Threshold and Correlation Analysis with MOS Difference instead of MOS. IEEE Speech Coding Workshop, 1999.