appeared in Proc. of European Wireless 2002, pp. 170-176, Florence, Italy, February 2002

Design and Deployment of Link-Layer Boosters for Per-flow Improvement of QoS in Wireless Internet Access

Christian Hoene, Iacopo Carreras, Tianwei Chen, Adam Wolisz[1]

Technical University of Berlin – Telecommunication Networks Group (TKN) – Berlin – Germany

Ph.: + 49 30 314 23818, Fax: +49 30 314 23819, e-mail: hoene|chen|wolisz@ee.tu-berlin.de

ABSTRACT

The transport mechanisms of Internet have an inherent drawback, which is due to their generality. Internet has to support a broad range of services (data, audio, video, etc.) on top of a unique protocol stack. On the other hand, it is well known, that there is a high potential for QoS and efficiency improvement if joint source and channel coding are used. Utilization of this potential is especially attractive in the case of interactive multimedia transmitted over wireless channels. In this paper we present the Mobile Adaptation System (MAS), which supports the dynamic deployment of boosters for flow selective link level packet treatment in wireless Internet access. These boosters use the knowledge of the importance of individual packets, or relations among packets in order to optimise the link layer transmission. We discuss our approach using as example a speech property based (SPB) booster for improvement of the perceptual quality of Voice over IP transmission in Wireless LANs.

1. INTRODUCTION

Wireless Internet access is already available. Huge investments in third generation access technologies will hopefully help to improve its quality, speed and usability. The aim of the paper is to show how the Quality of Service (QoS) of wireless communication services can be enhanced by individual treatment of packets within selected data flows.

Commonly the requirements for transmission QoS for multimedia streams are artificially and imprecisely extracted from the applications and formulated as requirements on network or transport level in terms of throughput, delay, jitter as well as loss probabilities. Different network solutions are compared according to these requirements.

A more attractive approach consists in comparing different networking solutions by direct measuring of the *perceived* quality of multimedia applications (see e.g. [1]). Users might evaluate the *subjective* quality of speech and video, or measurement tools calculating the *objective* quality by simulating the user's perception might be utilized. Such tools include multiple metrics and are to be developed separately for each type of multimedia application. The network/transport level QoS influences the perceived quality, but the network metrics are in a non-trivial relation to the perceived metrics.

One example of direct usage of the perceived QoS are recent investigations by Sanneck, Long Le et al. [2], who have studied the impact of packet losses to the speech quality. They analysed modern frame-based codecs like the ITU-T G.729 and have found out that losses of different frames (referred to as voiced and un-voiced) have a different impact with respect to perceived speech quality. In fact frame losses during an unvoiced/voiced transition impair the quality most significantly. Another example is the transmission of MPEG video, which consists of three different types of frames. Each of these types has a different importance for the decoder, thus losses have different impact depending on the frame type (in fact loss of some of the frames implies that some other frames are useless to the decoder, and might be dropped in advance as well).

As wireless links are expected to be bottlenecks in the future mobile Internet, offering much lower bit rates than their wired counterparts and dynamically changing the quality (fading), the above considerations are especially appropriate in this case. Unfortunately enough, the Internet treats – at the moment – all packets in the same way. Furthermore the main stream of discussions toward extensions of the recent Internet model by some QoS support tends to assure specific treatment of whole flows. If individual packets within a single flow might be marked differently, there is still no clear way to define conditions like: increase the importance of the following packet in the stream after the previous one has been dropped.

On the other hand joint source and channel coding (e.g. images over wireless [3]) has attracted recently more and more attention especially for transmission in difficult environments like wireless. It is well known that under finite delay requirements and in case of varying channel properties, joint source-channel coding is superior to Shannon's proposed separation of source and channel coding [4]. This difference might be even more dramatic if limits on the computational capacity of portable end-systems will be taken into account. But joint source-channel coding violates essentially the Internet's layering principle.

In this paper we propose the usage of intelligent, flow specific, link layer protocols, which should utilize the knowledge application requirements in addition to their local knowledge of channel conditions, in order to optimise the perceived quality of multimedia services.

We apply the term *link-layer booster* for these protocols, as boosters (see [5]), which transparently enhance existing protocols, seem to be the most attractive way to implement our concept. By definition boosters do not modify the syntax or semantics of the

[1] This work has been partially supported by the IST Project MOBIVAS.

exchanged end-to-end protocol messages and they work transparently with regard to other protocols.

We argue that it is important to have the possibility to exchange such boosters dynamically, this becomes necessary as new applications, or new coding schemata – calling for different processing – are deployed.

In this paper we discuss in §2 an architecture and protocol stack that supports per-flow treatment of the wireless link with link-layer boosters. In §3 we introduce the *Mobile Adaptation System* (MAS), which allows the dynamic deployment of boosters to re-configurable terminal in order to cope with changing applications and environments. In §4 we describe the design and implementation of a *Speech Property Based (SPB) Booster* that improves the quality of voice over wireless LANs. It uses characteristics of human speech production and features of modern audio codecs to differentiate packets regarding their importance for perceptual quality. We present our experimental measurement results of both the SPB booster and the Mobile Adaptation System. Finally we complete the paper with a summary of the results and an outline of further research.

2. ARCHITECTURE

For the sake of simplicity in this paper we will constrain our considerations to a wireless Internet access scenario with a single hop wireless link as illustrated in figure 1. A mobile terminal can use different wireless access networks to connect to the Internet. It communicates with corresponding hosts over the Internet backbone. Typically the throughput of the Internet backbone, and the wireless link differ significantly, for example by the order of a magnitude, because of different physical limitations of the underlying mediums. Thus, the wireless link has the largest impact on the end-to-end QoS. One should be aware, that the typical Internet approach to improve QoS by overprovisioning is very unattractive in the case of wireless links, because of the high cost of spectrum licencing, as demonstrated for example during the recent UMTS frequency auctions.

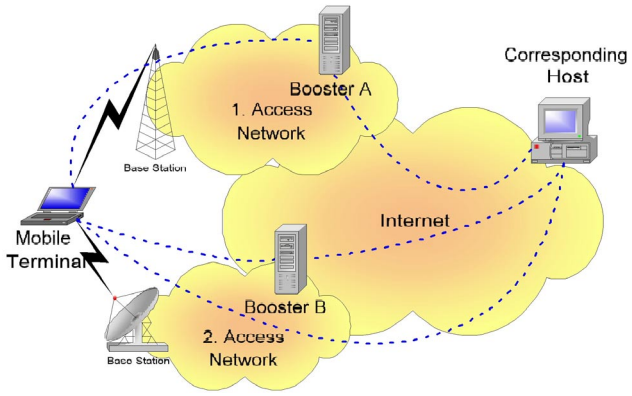

Figure 1: Wireless Internet access

To provide per-flow treatment of QoS over wireless links, the logical architecture of both mobile terminal and access point have to be extended. We describe here the supporting architecture for the mobile terminal, having in mind that the access point has to be designed correspondingly.

Four additional building blocks are introduced in the protocol stack of the mobile terminal (Fig. 2):

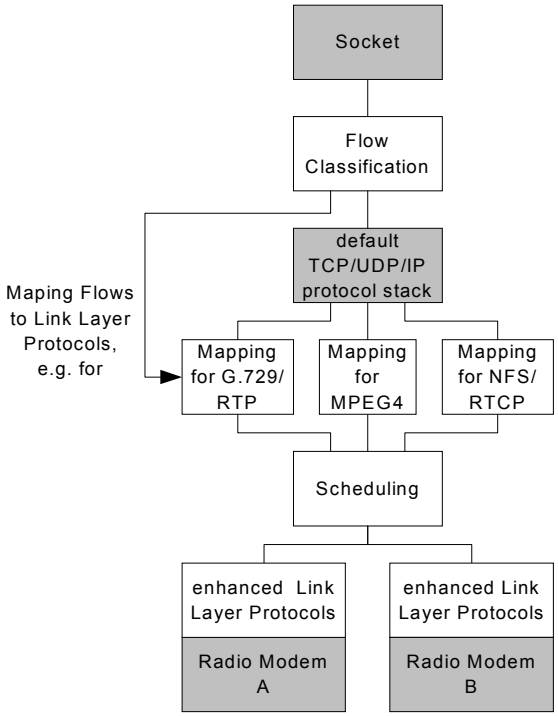

Figure 2: Transport plane on mobile terminal. The white boxes are either new in our architecture, or substantially extended in functionality.

2.1 Classification

A classification module identifies the flows of multimedia applications. The flow classifier uses information from the transport and application layer (there are different ways to extract such information – a good example might be found in [6]). It is placed below the Socket API and above the Internet protocol stack. Whereas we assume that the flow classification is autonomous, the introduction of QoS sockets [7] will extend the classification unknown applications. However, QoS sockets are still neither widely deployed nor used.[2]

2.2 Mapping

Flows are individually mapped to specific link-layer protocols, processing the stream of data with respect to the semantic knowledge about its structure (see e.g. §4).

2.3 Scheduler

Multimedia applications produce multiple flows thus a scheduler is always necessary in order to mediate their access to the shared transmission channel (as represented by an individual radio-modem). Introducing the possibility to treat individually packets within a single flow, as well as dynamic adaptation to the channel state opens new requirements for the schedulers. Simple FIFO schedulers, like used recently, will be substituted by much more complex ones.

[2] On the access point classification algorithms of flows are more complex. Their detailed description is beyond the scope of this paper.

2.4 Enhanced Link Layer

The transmission conditions on a radio channel vary substantially; there are also several possibilities to influence the quality of the transmission of the radio channel. Let us mention here only some of them, such as adjusting the transmission power, usage/non usage of multiple antennas, etc. Those special features have to be accessed over a specific common device driver API. We propose such API, following the approach presented in the Rooftop Data-Link API [8].

3. MOBILE ADAPTATION SYSTEM

The traditional Internet protocol stack is general enough to support a broad range of applications over heterogeneous networks. The introduction of new communication services or protocol enhancements is rather seldom. Furthermore, it takes a long time to create new ones, as it can be seen in the case of the ongoing deployment of IPv6. The link-layer boosters are designed to support a specific combination of multimedia application and wireless bearer. This combination is a priori unknown, because users install applications dynamically and the networking environment changes (at least due to terminal mobility). Thus the link-layer boosters have to be updated from time to time.

We argue, that network operators are responsible for deployment of the link layer boosters, because they have an immanent interest to increase the efficiency of the access network as well as an interest in competing in the offered QoS. On the other hand it is obvious that the majority of users do not want to actively bother about configuration of their terminals. Thus, our architecture bases on the assumption that the protocol update in the mobile terminal is triggered and executed by the networking operator (responsible for the access network).

3.1 Design

Traditionally, active or programmable networks have been proposed to accelerate the introduction of new communication services and to cope with the pace of network evolution [9]. Programmable networks abstract the networking hardware and include environments to download and install the networking software above the hardware abstraction. First of all, programmable networks usually assume the existence of a common abstraction of the underlying hardware. Unfortunately such an abstraction is not available yet for mobile terminals. One effort for creating such abstraction has been done within the DAPRA Rooftop APIs [8] describing a functional framework of radio APIs for various layers. This framework supports a broad range of different radio technologies, but an abstraction of the MAC layer, which link-layer protocols have to use, is missing. In addition this approach is not really disseminated. The Software Defined Radio Forum proposes the separation of MAC and link-layer in the Joint Tactical Radio System (JTRS) [10]. However, a detailed description of the API does not exist so far. Thus, proprietary interfaces have to be used, which differ for each radio modem type and often are tightly coupled to dedicated hardware. (An overview of link-layer APIs for wireless LANs can be found in [11].)

On the other hand programmable networks come along with a rather significant overhead because of both the abstraction discussed above, and dedicated execution environments. On a device with constrained resources this overhead might overwhelm the benefit of the link-layer boosters.

Our re-configurable terminal is based on an approach called *dynamic software updating,* as introduced by Hicks [12]. Protocol modules are downloaded to update or extend the existing code. Whereas dynamic software updating reduces the priori assumptions of the functions and allows efficient implementations, the questions when, what and from whom to download the code have to be solved.

Because boosters provide a special type of communication services, we use a service discovery mechanism to locate the booster (Fig. 3). First service providers (boosters) register their communication service at a service directory, which stores all communication services of an access network. Next, the client (mobile terminal) and the service directory try to find each other. The directory provides the client the description of an appropriate communication service. This information comprises the kind of service, a download address (URL) of the code to support the service and optional service configuration parameters. The client can download the code of adaptation agent (AA) and starts its.

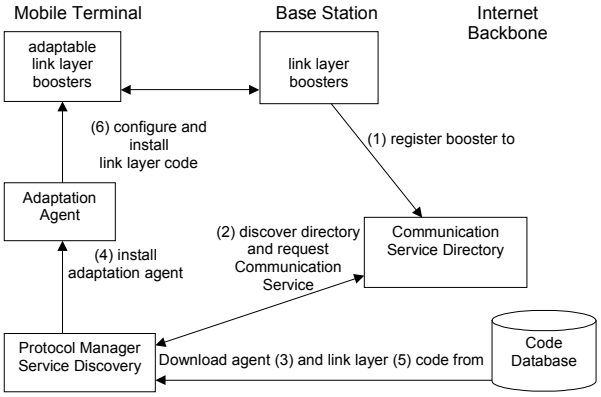

Figure 3: Mobile Adaptation System

To download the link-layer protocols, we follow an agent-based approach, described by Hall for the post-development configuration management of software development [13]. The AA configures, downloads, installs link-layer protocols. If it is necessary the client re-installs former versions of the link-layer protocols.

Three main building entities, the service discovery, the Adaptation Agent, Link Layer execution environment are described in the following.

3.2 Service Discovery

To implement the service discovery we have decided to use the IETF service location protocol (SLP) [14], because of its simplicity and increasing popularity. SLP comprises service agents, directory agents and user agent, and exchange and stores description of services.

A set of parameters are used to identify communication services. We used the following fields. The SLP type field describes the service class, e.g. “wireless/linklayer”. Two parameters specify the agent’s interfaces. (e.g. “java.net.ProgramDevDriver” for an

interface that a agent uses). An other parameter describes the execution environment for the agent: “downloadable protocols version 3”. At last we included some human readable descriptions of the communication service.

An user agent on the mobile terminal tries to discover a directory agent by sending IP multicast messages or the directory agent announces himself by sending a multicast message. After locating a directory agent the user agent asks for a suitable communication service and requests the downloading location of the upgrade agent.

The directory agent decides, which communication service is most suitable for the mobile terminal, if multiple boosters are available. Our service discovery framework is general enough to support different strategies to choose the right upgrade agent. Afterwards the download location, which is an Internet URL address, and optional a protocol configuration is sent to the user agent.

3.3 Adaptation Agent (AA)

The client downloads the code of the adaptation agent from code database, whose Internet address (URL) has been obtained with the service discovery.

To cope with various problems of mobile code technologies, we have decided to use the Java framework. The AA runs in an execution environment, which is based on the enhanced Sandbox of Java 2. Thus the agent is portable, save, secure. It can run on various hardware platform and operating systems.

Furthermore Java includes mechanisms to download code efficiently and on demand. The Java Archive (JAR) package supports the transfer of code (and other data) in a compress format [15] and as a single JAR file. Furthermore it can verify the integrity and origin of code by checking code signatures, which are part of a JAR file. Thus providers of code can assure the system stability. Their trusted code can run even outside of the secure sandbox.

One task of the AA is to identify the operating system, its version and the type of radio modem. It uses this information to download and to install a platform-specific link-layer protocol.[3]

3.4 Link-Layer

As the link-layer booster includes code influencing directly the radio modem, we have decided to include the link-layer booster in the device driver of radio modems. In a Linux system device drivers are part of the kernel. The Linux kernel supports an extension mechanism [16], which is used to exchange kernel modules (e.g. device drivers) at run-time.

The PCMCIA card manager [17] loads dynamically device drivers, if a new PCMCIA card is inserted in a slot. The card manager looks up the name of the PCMCIA card to find a suitable device driver and configuration scripts. Next, it executes the configuration scripts and installs Linux kernel modules. If a module cooperates with other modules, this module dependency is resolved automatically and all needed modules are loaded in the right order. If the PCMCIA card is a networking card, the card manager sets the right network configuration or triggers a DHCP client to request for an IP address.

In our design we have decided to retrofit this PCMCIA card manager for the installation of new device drivers and link-layer protocols. We have extended the card manager to support *name spaces,* which describe different configurations. If the adaptation agent requests an exchange, the card manager simulates the removal of the wireless PCMCIA card to unload the current driver. Afterwards a new name space is set and the card manager loads the newly downloaded configuration scripts, module dependency files, and kernel modules. Next the injection of the PCMCIA is simulated to reinstall and configure the new device driver.

3.5 Performance Measurements

For a proof of concept we have implemented our design and conducted some performance measurements. A notebook with a Pentium II 400MHz running LINUX 2.2.19 has been used as a re-configurable terminal. It was connected over a wireless LAN with an access point emulated by a standard PC. No other applications could use any of the resources during the measurements. On the access point we have placed the booster, the directory agent and the code server on the PC emulating the access point.

We are interested in the efficiency of booster discovery, downloading and installation. In table 1 the communication overhead due to downloading is shown while in table 2 the time scale of the update process is presented.

Action	User data in bytes	Network packets	Duration in ms
SLP: DA discovery	123	2	11,3
SLP: service request	389	2	29,3
Agent download	2971	14	64,5
Device driver download	45822	58	277,4
Device driver exchange			8680,6
DHCP	628	3	6,3
ARP	56	2	0,1
Total	**49989**	**81**	**9069,5**

Table 1: Network traffic due to booster download

Action	Start time in ms	End time in ms
SLP: DA discovery	0	11
SLP: service request	8614	8643
Agent download	8725	8790
Device driver download	8795	9072
Device driver exchange	10403	17380
DHCP	17380	17387
ARP	17407	17407

Table 2: Time scale of booster download

It is probably most interesting to look into the details of device driver exchange mechanism as it takes quite a

[3] The java environments can be used for transport- and network-protocols, too. However, the protocol performance is too slow to be usable.

long time, during which no network connectivity is available. In table 3 we have listed the actions of the card manager during the exchange. Most time is wasted during the simulated removal and insert, because the card manager has to run external programs and scripts.

Action	Time scale in ms
Start	0
Identification of PCMCIA card	0,3
Simulated removal	0,4
Change of the name space	3247,8
re-loading configuration files	3248,0
Simulated insert	3259,2
Finished	8681,0

Table 3: Exchange of link-layer protocol

On the other hand the gap between the DA discovery and the service request, due to implementation details, is less critical, as during this time network connectivity with the "old" software is available.

4. THE SPEECH PROPERTY BASED BOOSTER

In the following we will summarize the design, implementation and performance measurements of a speech property based (SPB) booster which improves the voice transmission of G.729 coded speech over IEEE802.11 wireless LAN in the case of high loss rates (more details can be found in [18]).

The basic idea of this booster is based on the observation that not all the packets have the same importance for objective speech quality [2]. Considering the speech signal properties and low bit rate codecs features, it is possible to distinguish between important and unimportant packets.

In order to evaluate the performance of the booster mechanism, we have set up an experimental measurement environment. The speech audio quality of the normal VoIP architecture has been compared with the speech audio quality using the novel booster mechanism. Because we are interested in the quality of the perceived speech audio, an objective measurement method has been used. The adopted metric is the perceptual distortion (EMBSD) [19], which yields results that have a high correlation with the MOS, the metric used in the case of subjective, human based tests.

Voice quality is reduced by packet losses even though common voice codecs include concealment algorithms, which interpolate lost audio segments. Packet losses can occur due to errors on the wireless link and late arrivals. We use a simulated play-out buffer to limit the maximal transmission time of voice frames. The play-out buffer drops packets that are too late. It is assumed that sufficient bandwidth is available to avoid losses due to congestion and queuing delays in the backbone.

We have experimented with three approaches providing an improved treatment of important packets[4]:

- *Selective packet loss recovery* protects packets by using different configurations of the physical and data link layer and switching between them on a per-packet basis. We have simply changed the maximum number of link layer packet retransmissions in case of transmission error. This is in compliance with the IEEE 802.11 standard.
- The second approach, called *redundant transmission*, protects the packets classified as important by performing a redundant transmission. The booster mechanism is responsible for duplicating several times the important packets. The Internet protocol stack has no restrictions regarding the duplication of IP packets.
- The third approach, referred to as *hybrid solution*, protects the important packets by both adding redundancy and using a selective packet loss recovery procedure.

4.1 Design of the SPB booster

In figure 4 the architecture of the booster mechanism is shown in the case of mobile terminal sending to the corresponding host. The booster is composed of a transmitting side, located on the mobile terminal, and a receiving side, located on the access point. Both parts of the booster are located at the MAC- and link layer. The part located on the mobile terminal is responsible both for classifying the packets (the "analysis" block) and for providing this information to the booster. Consequently, the MAC/link layer adopts or does not adopt a protected transmission, depending on the classification of the packet. The part located in the bridge is responsible for making the booster mechanism transparent to the rest of the system.

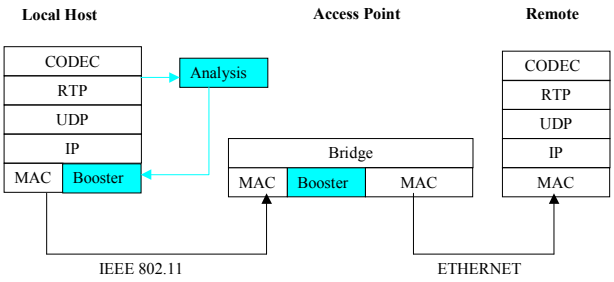

Figure 4: Speech property booster

4.2 Measurements and Evaluation

We have evaluated the SPB booster using an experimental set-up with commercial wireless LAN equipment. The measurements have been taken in our office environment using a laptop that has been connected over wireless LAN to a base station, and an corresponding host. We transmitted speech samples

[4] Several approaches to improve the Voice over IP have been already studied, let us mention just two of them: For Voice over IP Bolot [20] has proposed to use an open loop error control mechanism based on forward error correction. He argued that ARQ mechanisms are not acceptable for interactive audio applications because they dramatically increase end-to-end latency. The discussed solutions are only end-to-end. Bakin [21] has developed an FEC Booster for UDP applications over terrestrial and satellite wireless networks. The booster is similar to our redundant solution, but protects multiple consecutive packets by additional parity packets. Therefore, it increased the latency, because the lost packets can only be reconstructed after receiving all other packet in a FEC block.

using Java based RTP applications and a modified wireless LAN driver. We have measured packet losses, transmission delay, and both objective audio quality and packet losses.

The following plots are based on 400 measurements, each 15 s long, conducted in the case of high error rates, occurring on the outer limit of the transmission range during movement of the mobile terminal. The efficiency of the above discussed approaches has been compared with the standard, unprotected Voice over IP architecture.

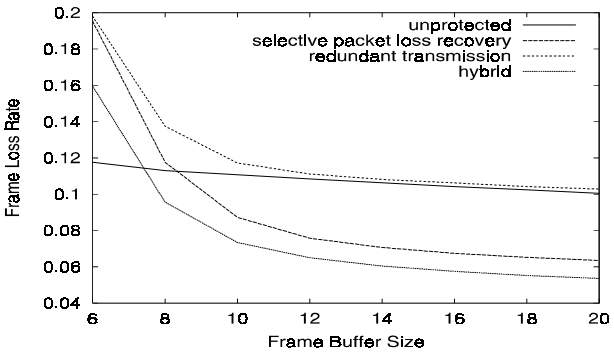

Figure 5: Frame losses vs. frame buffer size

In figure 5, the speech frame loss rate vs. the play-out buffer length measured in G.729 frames is presented (the scale has to be multiplied by 10ms to get the time dimension!).

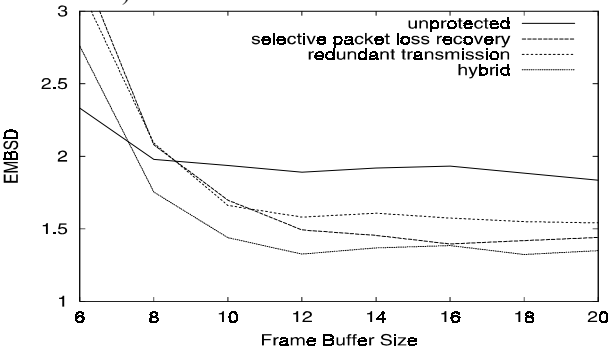

Figure 6: EMBSD vs. frame buffer size

In figure 6, the perceptual distortion (EMBSD) as a function of the play-out buffer size is plotted for all the investigated cases. Lower values of the index correspond to a better speech quality. Although ideally received samples would have a distortion of zero, due to the coding losses of G.729 lowest value of 0.9 have been observed if no transmission losses occurred.

Out of this sample results, it can be seen that utilization of the SPB booster leads in our experiments to an improvement of the audio quality. In the case of hybrid and redundant transmission these gains are due to a reduced loss rate (and come on the expense of additional channel load!). Contrary, in the case of selective packet loss recovery the voice quality is improved only by an intelligent distribution of the packet losses!

6. SUMMARY AND CONCLUSION

In this paper we have presented an architecture to improve the transmission mechanisms over wireless links. We propose the use of link-layer boosters, which take into account the multimedia application requirements and the channel properties.

We have discussed the design of a re-configurable terminal, which uses service discovery and upgrade agents to download and install these boosters. Unfortunately extended programmability comes along with additional resource consumption. Using a prototype implementation we have identified network traffic of 50 kbytes for the download and a service interruption of 9s for the exchange of the link-layer. As the protocol exchange should not be treated as a frequent event, we assess this values as promising for the first prototype. We are, however, investigating possibilities for their improvement.

A speech property booster has been developed and implemented as an example for the usage of our approach. Our experimental measurements have been successful in showing that the perceptual voice quality over wireless LAN can be significantly improved.

We are continuing our efforts on further optimisation of the SPB booster for voice transmission, in addition we are working on similar approach for MPEG-4 Video transmissions for wireless LANS. In addition we consider solutions for proper boosters also for other wireless technologies.

ACKNOWLEDGEMENTS

Authors would like to thank the colleagues from the TKN Group, especially Morten Schläger, as well as MOBIVAS partners for numerous motivating discussions. We would like to express our gratitude to Ing. Enrico Gregori (CNUCE Pisa) for having made the visit of Iacopo Carreras at Technische Universität Berlin possible. Christian Hoene thanks the Technical University of Berlin for providing a scholarship for a visit at Columbia University and Prof. Campbell for coaching him during this visit.

REFERENCES

[1] A. Watson and M.A. Sasse. “Measuring Perceived Quality of Speech and Video in Multimedia Conferencing Applications”, In Proceedings of the ACM Multimedia Conference, pp. 55-60, Bristol, UK, September 1998.

[2] H. Sanneck, N.T.L. Le, and A. Wolisz. “Intra-flow Loss Recovery and Control for Voice over IP”, Proceedings of ACM Multimedia, 2001.

[3] N. Farvardin and V. Vaishampayan, "Optimal quantizer design for noisy channels: an approach to combined source-channel coding", IEEE Trans. Inform. Theory, Vol. 33, pp. 827--838, Nov. 1987.

[4] C.E. Shannon, “A mathematical theory of communication”, Bell System Technical Journal, 27:379--423,623--656, 1948.

[5] D.C. Feldmeier, A.J. McAuley, J.M. Smith, D.S. Bakin, W.S. Marcus, and T.M. Raleigh, "Protocol boosters", IEEE Journal on Selected Areas in Communications, Special Issue on Protocol Architectures for 21st Century Applications, 16(3):437-444, April 1998.

[6] T. Harbaum, M. Zitterbart, F. Griffoul, J. Röthig, S. Schaller, H. J. Stüttgen, “Layer 4+ Switching with QoS support for RTP and HTTP”, Proceedings of IEEE

Globecom Conference, Rio de Janeiro, Brazil, December 1999
[7] Microsoft Cooperation, “Windows Sockets 2 Application Programming Interface, An Interface for Transparent Network Programming Under Microsoft Windows, Revision 2.2.2, August 7, 1997”, ftp://ftp.microsoft.com/bussys/winsock/winsock2/
[8] Rooftop API, www.sdrforum.org
[9] A. T. Campbell et al., "A Survey of Programmable Networks", ACM SIGCOMM Comp. Commun. Rev., Apr. 1999.
[10] Software Defined Radio Forum, http://www.sdrforum.org/
[11] Data-Link Programming, http://www-tkn.ee.tu-berlin.de/research/dp/
[12] M. Hicks and S. Nettles. “Active networking means evolution (or enhanced extensibility required)”, in Proceedings of the Second International Working Conference on Active Networks, October 2000.
[13] R.S. Hall, D.M. Heimbigner, A. van der Hoek, and A.L. Wolf, “An Architecture for Post-Development Configuration Management in a Wide-Area Network”, in Proceedings of the 1997 International Conference on Distributed Computing Systems, pages 269--278. IEEE Computer Society, May 1997.
[14] E. Guttman, C. Perkins, J. Veizades, and M. Day. “Service Location Protocol, Version 2”, IETF RFC-2165, Nov. 1998.
[15] Sun Microsystems, Inc. Java Archive (JAR) File. http://www.javasoft.com/products/jdk/1.2/docs/guide/jar
[16] Matt Welsh, “Implementing Loadable Kernel Modules For Linux”, Dr. Dobbs Journal, 20(5), 1995.
[17] Linux PCMCIA Information Page, http://pcmcia-cs.sourceforge.net/
[18] C. Hoene, I. Carreras, and A. Wolisz, "Voice Over IP: Improving the Quality Over Wireless LAN by Adopting a Booster Mechanism - An Experimental Approach", in ITCOM 2001, Denver, USA, August 2001.
[19] W. Yang, M. Benbouchta, and R. Yantorno, “Performance of the modified spectral distortion as an objective speech quality measure”, in Proc. ICCASP, vol. 1, Seattle, Washington, Mai 1998.
[20] J.-C. Bolot, and A. Vega-Garcia, "The case for FEC-based error control for packet audio in the internet", ACM Multimedia Systems, 1997.
[21] D. Bakin, W. Marcus, A. McAuley, and T. Raleigh, “An FEC booster for UDP application over terrestrial and satellite wireless networks”, International Mobile Satellite Conference (IMSC 97), Pasadena, California, June 1997.
[22] L. Muños, M. García, J. Choque, R. Agüero, and P.Mähönen, “Optimizing Internet Flows over IEEE802.11b Wireless Local Area Networks: A Performance-Enhancing Proxy Based on Forward Error Correction”, IEEE Communication Magazine, pp. 60-67, December 2001