White Paper: Lightweight Protocol for Hardware-Efficient DVB Distribution in IP Networks

GEORG ACHER, DETLEF FLIEGL
BayCom GmbH
and
THOMAS FUHRMANN
Technische Universität München

1. INTRODUCTION

In this paper we present NetCeiver, a lightweight and energy-efficient IPTV architecture that is fully compatible to existing digital television broadcast systems. It allows an easy convergence of digital television and IPTV. During the transition, consumers can thus benefit from the advantages of both technologies.

NetCeiver pursues an integrated approach that combines zero-configuration algorithms with hardware-software co-design to achieve a seamless convergence of DVB and IP television. Throughout this paper we use the term *convergence* to describe a way of delivering content independently of a specific transport medium. In traditional broadcast systems this would be called simulcast. IPTV allows convergent deliverance of television content over IP.

NetCeiver is based on reconfigurable hardware. Thereby, it achieves a great flexibility with respect to upcoming standards such as DVB-IPTV. Moreover, NetCeiver is fully IPv6-compliant to ensure compatibility with future network requirements.

In the following, we first briefly introduce some typical characteristics of traditional television broadcast systems. Based on our analysis we derive a novel architecture for transparently transporting broadcast TV services over IP.

2. DVB TO IP NETWORK BRIDGING

2.1 Digital Television Broadcast Systems

Despite the fact that IPTV systems have been available on the market for quite some time, legacy distribution systems for digital television are still predominant. Most TV stations employ these systems, and as a result, they are more popular than any other IPTV solution – at least in western countries.

Author's address: G. Acher, D. Fliegl, BayCom GmbH, Guardinistr. 47, 81375 München, Germany, `{acher|fliegl}@baycom.de`
T. Fuhrmann, Lehrstuhl für Netzarchitekturen und Netzdienste, Fakultät für Informatik, Technische Universität München, 85747 Garching bei München, Germany, `fuhrmann@net.in.tum.de`

These legacy distribution systems broadcast according to well established standards over terrestrial antenna, satellite transponders, or broadband cable networks. Inexpensive receivers provide consumers easy access to these services. Typically, their quality exceeds that of most current IPTV implementations.

Digital broadcast distribution systems operate according to standards such as DVB [ETSI] , ATSC, ISDB, DMB. For simplicity we refer to all these systems as DVB, because they all use the MPEG2 transport stream format (MPEG2-TS) for their transmissions.

Digital television is wide-spread. Consumers accept it thoroughly. DVB provides high quality audio and video transmissions, fast channel switching, and a vast number of additional services such as electronic program guides or conditional access (Pay-TV). Many of those services are fully standardized. As a result, consumers can trust that any commodity receiver is compatible with their preferred service provider. As stated above, this situation is still totally different in current IPTV systems.

If we had the same features for IPTV, we could rely on well standardized client applications or set-top boxes that we could use with more than one IPTV service. Unfortunately, this is unlikely to happen in foreseeable future, because IPTV systems have to meet too many different, mostly commercial interests.

In this paper we present a possible solution: Our NetCeiver approach is based on DVB. It makes all features of DVB broadcast systems available from within IP-based local area networks. In the next sections, we present the NetCeiver architecture, in particular its communication protocols and hardware details.

2.2 Mapping of DVB to IPTV

The idea of delivering digital television broadcast to IP networks is not new at all. There already exist several implementations that receive, transcode, and distribute TV programmes via IP. All these solutions provide random access to audio and video streams, which enables the user to watch TV on LAN attached PCs or set top boxes. But a closer look at these implementations reveals that they do not transparently bridge all digital broadcast features [Djama and Ahmed 2006]. Typically, they rely on pre-configured channel and service lists. Therefore only a fixed set of additional services such as the electronic program guide or teletext is converted for IP usage. Furthermore, these systems do not keep the broadcast character of the received streams. Rather they use unicast transport connections.

The recently upcoming standard DVB-IPTV [DVB Consortium 2009] is an exception to this. It reflects the broadcast character of DVB, but is not yet widely accepted for production use and requires an expensive infrastructure. Even tough DVB-IPTV standardization is not yet finished the framework currently consists of several complex technical specifications. For example DVB service information requires costly conversion into XML data structures. Also single PIDs of a DVB service have to be rearranged into an IP stream which is not bandwidth efficient for services with shared PIDs.

2.3 The NetCeiver Architecture

The primary goal of the NetCeiver architecture is to map full DVB services to IPTV without changing their quality, feature set, or their broadcast character.

It should be possible to access digital television services without prior (network) configuration, proprietary client software, or proprietary set top boxes. Multiple DVB services, potentially received on different frequencies, should be available for multiple clients in the local IP network. The latter has be to achieved in a scalable manner.

In this paper we describe how the NetCeiver achieves these design goals. As depicted in fig. 1, the NetCeiver can be seen as a head end for DVB distribution within IP based networks. The NetCeiver system provides a fully convergent solution in terms of our definition above. There are no restrictions when using DVB services from an IP network.

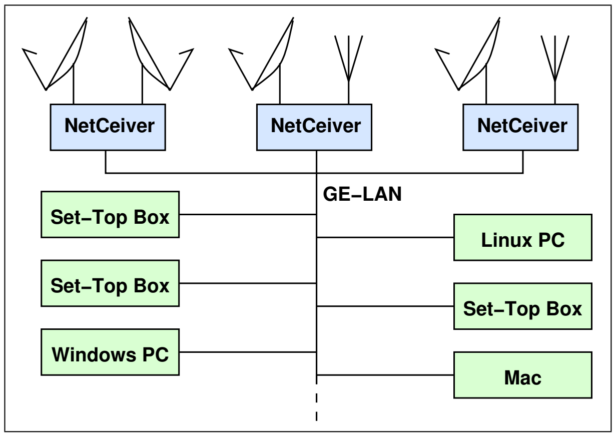

Fig. 1. Usage scenario

The NetCeiver architecture is based on highly specialized, cost efficient hardware with a low power requirement. It provides slots for DVB tuners, CAMs and has a Gigabit Ethernet interface (see below). It runs a Linux based operating system, which allows resource efficient, simple and flexible software development.

2.4 Software Architecture

As stated above, we designed the NetCeiver system as an IPTV system for DVB services. For this purpose we had to first select a suitable transport protocol. Many multimedia streaming frameworks already exist and mostly deliver compound streams of audio and video to a client using an unicast protocol. Later in this section we will discuss some of these frameworks more in detail. To find out which framework is suitable for our purpose we are now going to define our requirements:

(1) Transparent DVB access: Support for all quality modes, the entire DVB feature set, and low zapping times (below 1-2 seconds).

(2) Zero configuration: A self-organizing system that runs without any prior setup.

(3) Scalability: Multiple servers can serve multiple clients in an efficient way.

(4) Low cost solution: Server and client are designed to meet low cost requirements.

The rationale for these requirements is based on the way how digital television broadcast systems operate: DVB services are organized as a multiplex called *transport stream* (TS). Each TS contains a number of logical channels, which are identified by a unique 13-bit value, the *Packet Identifier* (PID). The so-called *Program Map Table* (PMT) lists and describes all PIDs that belong to a DVB service. Therefore it is necessary to selectively access single PIDs of a TS. A DVB receiver can bootstrap all available services (PMTs) by accessing the *Program Allocation Table* (PAT). The PAT is always located on PID 0.

Studying various IPTV and streaming frameworks we learned that they do not well support this DVB operation scheme:

(1) None of the established protocols or streaming framework efficiently satisfies the requirement of transparently addressing single PIDs of DVB transport streams.

(2) Some existing protocols and frameworks, such as UPnP-AV [UPnP Forum 2008] (also DLNA), Multicast DNS [RFC Draft 2008] or SLP [IETF 1999] provide zero configuration support, but they are focused on already pre-assembled services. Implementing them for the NetCeiver system might be possible to some extent, but it would not be fully compatible with the desired operation scheme. Other platforms [Fabio Forno et al. 2006] are based on complex frameworks like MHP and cannot be implemented in low cost designs.

(3) The system has to be bandwidth efficient to be scalable: When different clients request the same PID, the corresponding data still should be sent over the network just once. This requirement can only be met by multicast capable streaming protocols [Sun et al. 2007; Ma and Shin 2002].

(4) Solutions like the standard DVB-IPTV require a costly server infrastructure, where many different services and protocols have to be supported. On the client side, traditional DVB-only set-top boxes need significant software changes. It is not easily possible to just replace the integrated tuner by a network data source.

2.4.1 *IPv6 Multicast based DVB Access.* In order to meet these requirements, we designed the NetCeiver to selectively stream single PIDs of a DVB transport stream into a LAN using IPv6 multicast. Each stream is controlled by the standard IPv6 multicast listener discovery protocol (MLDv2 [IETF 2004]). This standard describes how to report joined or recently left multicast groups within a network segment: An IPv6 multicast capable router periodically sends out a multicast listener query to discover the presence of multicast listeners. Every multicast capable network node that has joined or recently left at least one multicast group reports its list of groups to the querier. Furthermore, a multicast node reports joining or leaving groups immediately without a prior listener discovery.

The NetCeiver exploits the MLDv2 protocol by sending out multicast listener queries periodically and receiving all multicast listener reports. This results in an up-to-date list of all joined multicast groups. Now the NetCeiver needs to know which PID from which transponder or frequency has to be streamed on a given multicast group. To achieve this it applies an algorithm that transforms specific receiver settings into an IPv6 multicast group and vice versa.

An IPv6 address consists of 128bit, which are written with hexadecimal characters in eight groups of 16bits each [IETF 2006]. Depending on the application there exist different address schemes. This means that not the entire 128bit address space can be used freely. For example, multicast applications have to set the first 8bits to 0xff, followed by a 8bit scope identifier. Only the remaining 112 bits differentiate between the multicast groups.

When a specific PID of a DVB transponder is selected, our algorithm calculates a corresponding 112bit multicast group address. The so calculated multicast group uniquely identifies the tuning parameters. Hence, we can also calculate the inverse mapping. Moreover, 112 bit contain enough space to identify all relevant parameters: a delivery system (satellite, terrestrial, etc.), orbital position (for satellites), frequency, polarization, symbol rate, FEC modes, and PID. Fig. 2 gives an example of this scheme. Thus a client tune request is issued just by joining a matching multicast group which contains all relevant information. Unlike in typical multicast usage, this group is not required to exist or to be announced, its service and the streaming resources are created *on-demand* and set-up *after* the client request.

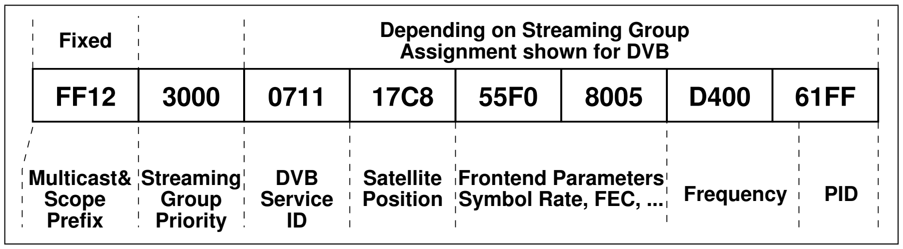

Fig. 2. IPv6 Multicast Group Decoding (example for DVB)

2.4.2 *Zero configuration protocol.* So far, we have just presented a transparent multicast based mapping of DVB parameters. The next requirement deals with the zero configuration feature of the NetCeiver system. To this end, we defined a fixed set of multicast groups. All clients and servers have to join these groups, independent of any DVB stream (cf. fig. 3). All NetCeivers distribute their hardware capabilities and their current tuner allocations on these groups periodically. We use XML coded data structures for this purpose. They comply with the W3C RDF standard CC/PP (Composite Capability/Preference Profiles [W3C 2004]).

Based on this information, a client can find available NetCeivers and configure itself automatically accordingly. When a client joins a multicast group for receiving a specific PID, it also joins a corresponding group that distributes the signal information from the tuner in charge.

When there are multiple NetCeivers on a network segment, it is also necessary to synchronize all NetCeivers so that they have consistent information about their state. This ensures a proper handling of requests if more than one NetCeiver could serve a multicast group. It also enables resilience in case one NetCeiver fails.

All of these features work in IPv6 without any prior network setup because every IPv6 capable network interface automatically receives a link local address. With such an address is it possible to communicate within a network segment.

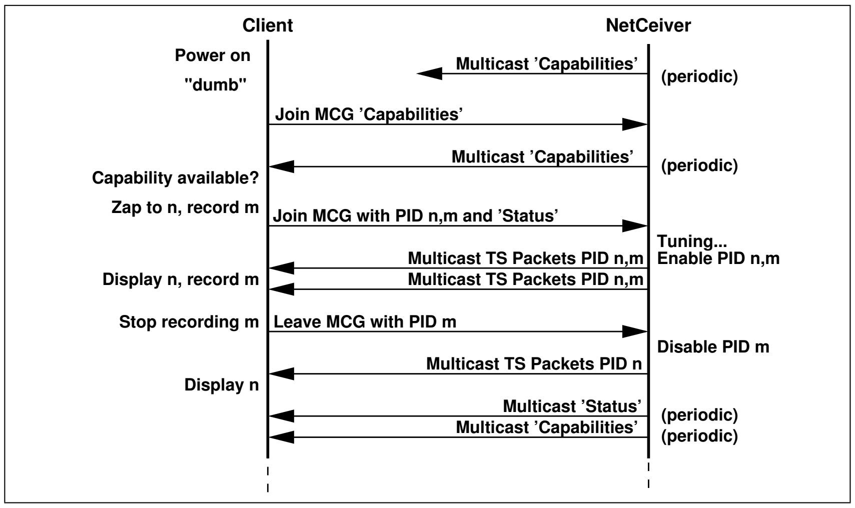

Fig. 3. Zero configuration and implicit controlling via join and leave

2.4.3 *Scalability.* The last requirement remaining is scalability. As said above, we wanted to have a resource and bandwidth efficient system that can handle multiple servers and clients. The unique multicast group calculation algorithm prevents the system from duplicating streams for different clients. The self-organization features allows a potentially large number of clients to concurrently feed from one or multiple NetCeivers.

Given the bandwidth limitation of a gigabit network, it is possible to stream a large number of DVB services: About 100 HDTV or 160 SDTV programmes[1] could be streamed simultaneously. A small number of NetCeivers suffices to satisfy such a high demand: Each NetCeiver can stream six full DVB-S2/DVB-C transponders, leading to a maximum bandwidth of about 300Mbit/s. Given a typical mixture of requests, the number of NetCeivers should not exceed five, unless the switching hardware can snoop MLDv2 and thus limit the traffic to the respective clients.

Currently, there exists a restriction when accessing a NetCeiver beyond router boundaries. Due to the operation scheme of multicast routing [IETF 1998] it is impossible to propagate join operations for groups under certain conditions: Traffic on multicast groups is being generated by the NetCeiver upon an incoming join request of a client. A router cannot determine where to forward this request for a yet inactive group of a specific NetCeiver.

2.5 Server Implementation

The NetCeiver server runs a Linux based operating system which has many Open Source libraries and services already available. Thus the server functionality relies on well known APIs of libxml2 and Pthreads. Because the DVB subsystem of the NetCeiver is implemented in highly specialized hardware there is no need for DVB support in the Linux kernel. The server is organized in two processes, one that

[1]Assuming a bandwidth of 8Mbit/s in H.264 for HDTV and 5Mbit/s MPEG2 video in SDTV.

operates as an hardware abstraction layer (HAL) and another one building the application server (see figure 4).

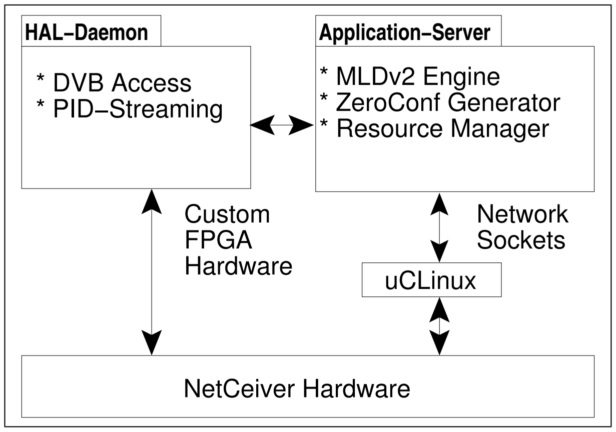

Fig. 4. Software components of NetCeiver server

The application server contains a MLDv2 compatible querier which collects and handles all multicast listener responses in realtime. Multicast groups that can be resolved into DVB PIDs are forwarded to the resource manager. Latter takes care for allocating DVB tuners by a preference function and finds out if a multicast group can be redirected to another one that already streams the same content. If an available tuner resource can be found in the system the DVB tuner is being accessed and hardware based streaming is enabled.

Another important feature of the application server is the zero configuration support. Therefore it collects internal hardware information and current resource allocation and transmits these data on predefined multicast groups. It also joins these groups and receives information from other NetCeivers in the network. This information being used for preference calculation of the resource manager. Thus it is possible to have multiple servers without prior configuration.

2.6 Client Implementation

The client functionality is provided by a portable library which can be used on Linux, Windows and MacOS. Because essential MLDv2 support is not available natively on Windows XP and MacOS the NetCeiver client implements its own MLDv2 reporter. This also could speed up zapping between channels under certain conditions.

The library API is split into zero configuration support and PID streaming functions. First a client application can find out which DVB sources are available on the network. Second socalled receivers can be instantiated which provide access to DVB transport streams and out of band information like signal parameters.

3. HARDWARE

3.1 Overview

Fig. 5 shows the NetCeiver hardware components. We have built a System-on-Chip (SoC) upon a Xilinx XC3S1600E FPGA [Xilinx Inc. b] with nominally about 1.6 million gates. No standard SoC provides the required performance and data handling capabilities, so we decided to use an FPGA. Although, in general, an FPGA is more expensive than a standard SoC, it is nevertheless the key for the NetCeiver's low component cost of less than EUR50 (excluding tuners). Moreover, the dedicated streaming logic also has a very good power efficiency, namely approximately 4-6W excluding tuners[2]. The FPGA does not require a heat sink so that the system does not need a forced cooling. This further increases its overall reliability.

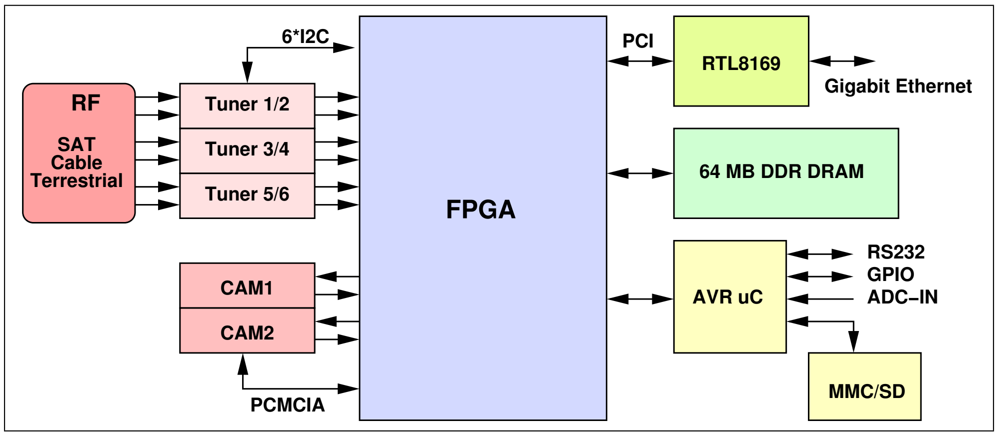

Fig. 5. Hardware overview

The FPGA interfaces to three exchangeable tuner cards. Each card can deliver the transport stream of two tuners. Thus a NetCeiver board can use up to 6 DVB tuners in parallel. The overall input bandwidth peaks at about 35 to 40Mbyte/s, depending on the tuned transponders.

Two slots for Conditional Access Modules (CAMs) provide the option to decrypt the received transport stream data on the fly.

Besides the FPGA, our NetCeiver design requires only a few active components: An AVR micro-controller with a MMC/SD-Card interface initializes the SoC with the FPGA bit-stream and the uCLinux image. After boot-up, the AVR allows simple block device IO to the flash card and acts as an IO expander.

The FPGA has access to a 64Mbyte DDR-DRAM. One half of the memory is used by the streaming hardware to buffer the TS packets, the other half is dedicated to the uCLinux-OS.

A PCI-connected Realtek 8169 Gigabit Ethernet interface handles the network connection. The RTL8169 provides a high performance (above 100Mbytes/s) for

[2]A DVB-S2 tuner consumes about 2-4W.

very low cost. This makes it an ideal interface that did not require much development work. Additionally, the chip has two separate transmit queues. This made it easy for us to implement an independent hardware controlled streaming in parallel to the operating system. We explain this in detail later.

A summary of the technical specifications is given in table 3.1, the PCB is shown in fig. 6.

Tuners	6 TS inputs (each up to 9Mbyte/s)
CAMs	2 (2 TS in, 2 TS out)
Network	RTL8169 Gigabit Ethernet
FPGA	XC3S1600
Internal clocks	66MHz except 33MHz PCI, 132MHz DDR & TS matrix
Memory	64MB DDR-DRAM, 128MB Flash
Power consumption	4-6W (excl. tuners and CAMs)
OS	uCLinux
Boot time	approx. 25s (depending on tuners)
Dimensions	approx. 28cm*12cm (16cm*12cm w/o CAMs)
Height	8cm (incl. network and tuner cards)
PCB	4 layer
Component Cost	<EUR50 (at mid volume quantities)

Table I. Technical Specifications for the first NetCeiver HW Implementation

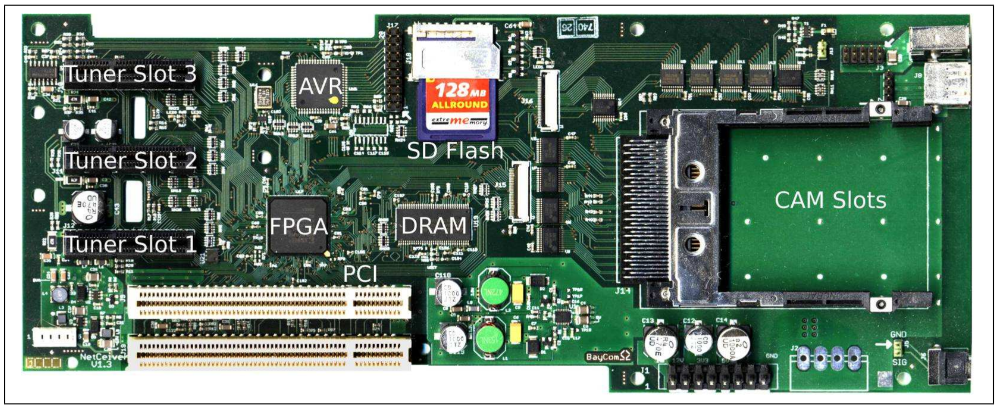

Fig. 6. PCB of the NetCeiver

3.2 FPGA data flow

Fig. 7 depicts the SoC that is implemented inside the FPGA. It consists of the CPU core, the memory controller, a PCI subsystem, and the TS and network stream processor. The CPU uses IP cores provided by Xilinx in the Embedded Development Kit (EDK), all other blocks were specifically developed for the NetCeiver.

The main CPU block is realized by a Microblaze IP core [Xilinx Inc. a] running at 66MHz. The CPU core has instruction and data caches with 8KB each. Although

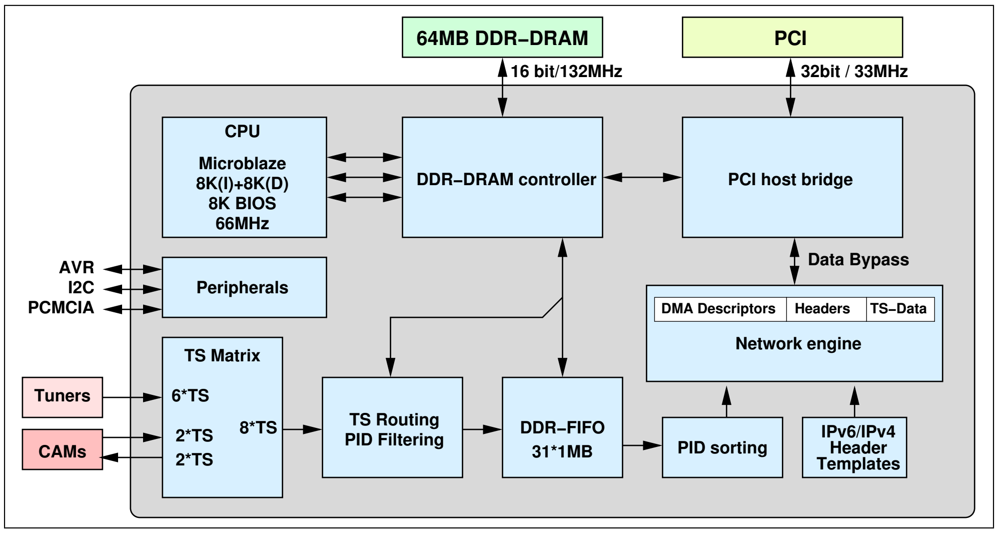

Fig. 7. Internal components of the FPGA

the Microblaze is a full featured 32bit RISC CPU and can run the MMU-less uCLinux, it requires only about 10% of the FPGA's resources.

The memory controller interfaces to a single-chip, 16 bit data bus DDR-DRAM clocked at 132MHz. Large buffers for each of the 5 internal ports allow a sustained memory bandwidth of about 400Mbyte/s for longer burst accesses.

The PCI subsystem implements a 32bit/33MHz host bridge (initiator/target) to connect to the RTL8169. This allows us to use the 8169 Linux driver without modifications. Because of the relatively slow CPU, the network bandwidth is only about 2-3Mbytes/s. This is sufficient for MLDv2 handling, but obviously not for streaming multiple DVB channels. They require a much higher bandwidth.

For this purpose, the PCI system contains a bypass data path. It is enabled when accessing a specific memory area. This area does not map to system memory but to special registers in the network streaming controller. The second transmit queue of the RTL8169 reads on-demand hardware-generated DMA descriptors and packet data over this bypass. This fully offloads the CPU from the packet generation process for streaming. A benchmark using synthetic data packets showed that the RTL8169 and the PCI-bypass achieved a transmit rate of about 100MB/s, which is about 80% of a Gigabit link capacity. This is fully sufficient for our purpose, even when the clients request six full DVB transponders.

On the input side, the TS data from the six tuners is routed through a matrix for CAM insertion. Afterwards, a table lookup based on the packet ID (PID) determines if at least one client requires this PID at all. This *PID-info table* then also determines the way the packet should take. For the main data path network streaming[3], FIFO areas (1MB per tuner) in system memory buffer the packets.

As already mentioned, one key element of the NetCeiver multicast protocol is that

[3]There are also some other FIFO destinations specified in the PID-info. NetCeiver supports a maximum of 31 FIFOs. They are used internally for monitoring and service information parsing.

each PID is sent over a separate multicast address. Unfortunately, DVB delivers the PIDs in a randomly multiplexed order. They are not sorted to form bursts of one PID that could fill up the maximum transfer unit of a network packet. Thus we need to store and aggregate the packets. For this purpose, we scan 32 successive TS packets in the FIFO and sort them according to the PID. This allows us – at least for the high bandwidth video PIDs – to better utilize the MTU. This ensures the desired high throughput. Typically, a network packet contains 1378 bytes, 62 bytes in the headers (Ethernet, IPv6 and UDP) and 1316 bytes in seven TS packets. The outcome of this aggregation is shown below.

The network data generation unit then assembles the full network packet from such aggregated TS packets. The basic IPv6 Multicast streaming header is based on templates that the software in the CPU core fills in advance. When the network chip accesses the header area over the bypass path, the data is finalized on the fly while being transferred. That means that the transfer unit inserts header fields such as the payload length or the destination IPv6 address into the template placeholders. After the header data it reads the packet data as is from the respective packet buffer.

Besides supporting IPv6 multicast, we made provisions in the PID-info table to enable IPv4 multicast and unicast, too. A unique ID, which can be propagated to the data generation engine, selects a linked list of individual connection data such as a destination MAC, an IPv4 destination address, RTP sequence counters, etc. This allows us to implicitly multiplex different PIDs to one destination address. As a result, the hardware also supports traditional streaming protocols and clients.

The described TS handling and network packet generation runs in parallel to the CPU. It does not need any CPU intervention. The CPU only configures the IP header templates and updates the PID-info tables for each started or stopped streaming request. Our tests confirmed that the hardware is actually capable of streaming the full content of six tuner transport streams (the mentioned 35-40MB/s) without problems.

All the SoC component occupy only about 50% of the FPGA so far. This leaves plenty room for future extensions.

4. PERFORMANCE MEASUREMENTS

The performance of the NetCeiver system can be measured in several ways. The most crucial impact is caused by the zapping latency. Thus we decided to present the zapping latency distribution of a real life scenario. The testbed consists of a system with 2 NetCeivers (equipped with 6 DVB-S Tuners) and 3 Linux clients. The measured latency is a sum of these single timing values: DVB frontend tuning, NetCeiver resource manager, Network and multicast join/leave, client operating system / DVB subsystem overhead.

The test is based on a modified szap utility, which is part of DVB applications for Linux DVB. The latency was measured from tuning to a random transponder until frontend lock and begin of streaming. For the graph shown in fig. 8 a number of approx. 1000 tuning requests was issued concurrently by the 3 clients. The graph shows exemplarily the histogram of one client.

The initial peak at approx. 70ms shows the number of tuning requests when a

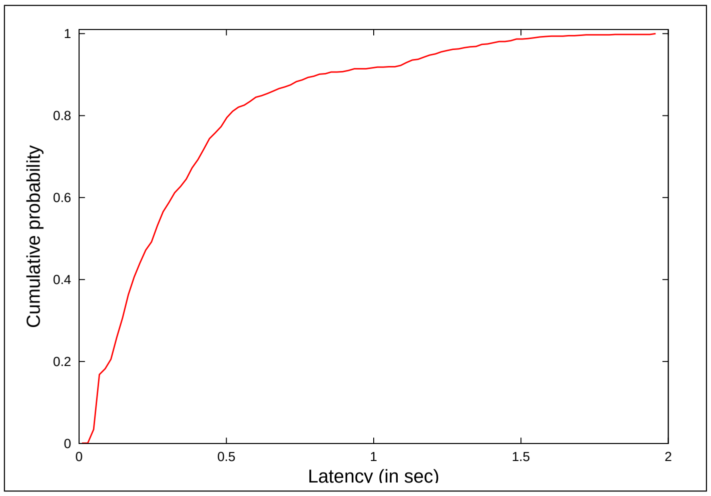

Fig. 8. Zapping latency in real life scenario

transponders was already locked on one of the DVB tuners. So there was no tuning latency. Most of the requests could be served within 0.5s and only few took up to 2s. The requests with a very long latency are thoroughly caused by the DVB frontend tuning latency. A missed multicast listener report causes an average latency of 1s which is the polling interval of the multicast listener querier. Even though the CPU of the system runs at 66MHz only, this does not contribute negatively, because the execution speed of the server process does not influence the measured latency.

We also examined the efficiency of the packet aggregation. Figure 9 shows the distribution of packet sizes for one typical TV service, including PIDs for video (about 8 MBit/s), audio and EPG. About 60% of all packets carry the maximum possible payload size. Audio PIDs appear only once or twice in the aggregation window, so about 60% of them cannot be combined. In summary, the overhead for the 62 bytes IP-header is about 11.2%.

Figure 10 shows the packet distribution for a different scenario. In 20MBit/s three TV services from one tuner are streamed. The video bandwidths are about 8MBit/s for two services and 3MBit/s for the third. Here, the 32 TS packet window is divided by the three video streams. The full packet usage drops to about 30% and smaller packets for video data appear more often. Still, the single TS packet is mainly used for the low bandwidth audio data. In this case, the header overhead is about 15.5%.

These results imply that the aggregation window should be increased when the NetCeiver is used in a DVB head-end scenario with many clients[4]. Since the current choice for 32 TS packets is based on the uncascaded depth of the FPGA's distributed RAMs, it is not fixed by the HW and can be easily increased. On the other hand, an aggregation over too many packets changes the realtime properties of the stream and maybe complicates instant audio/video sync.

[4]It should be noted that the aggregation capability is not affected when streaming from multiple tuners.

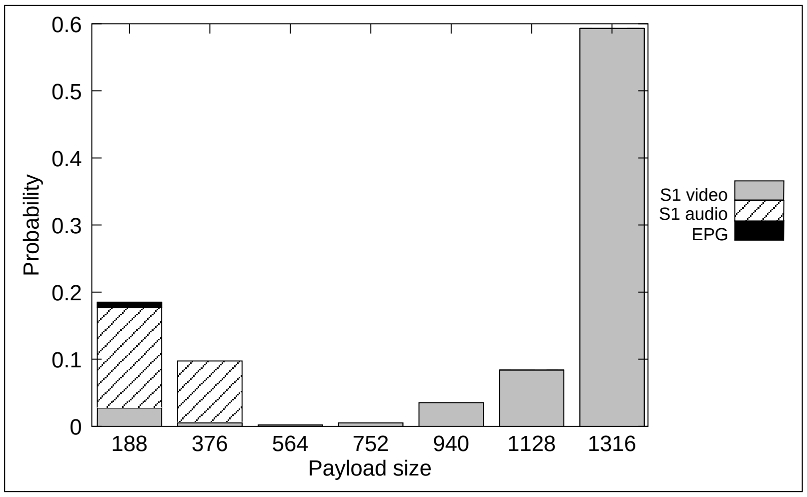

Fig. 9. Payload size distribution of a single service

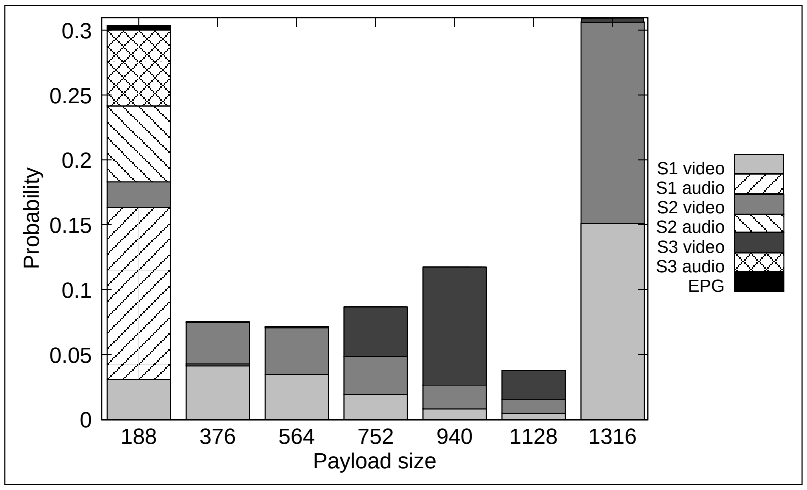

Fig. 10. Payload size distribution of multiple services

5. CONCLUSION AND OUTLOOK

In this paper we have presented the NetCeiver architecture for distributing DVB-like broadcast over IP networks. NetCeiver uses standard protocols such as IPv6 multicast with MLDv2. Our main design goals were zero configuration, scalable streaming to an arbitrary number of clients, and full service transparency for typical DVB receivers.

We have developed a cost-efficient FPGA based hardware that implements our NetCeiver architecture. It allows easy deployment in the consumer market. Moreover, our design is power efficient and allows a very high network performance. We achieve this with a specialized stream handling inside the FPGA.

The NetCeiver architecture is available through a commercial vendor since late 2007. The customer feedback shows that it meets its design goals. The NetCeiver product works as a detachable DVB tuner component of a high-end set-top box. It is also available as a stand-alone head end for inexpensive and lightweight STB clients.

Yet, the development is not finished. In the future, we will extend the NetCeiver core functionality by VOD-services. Furthermore, in the work presented here we did not consider security issues. All devices inside the network are supposed to be trusted. Future evolutions of the system should consider authentication methods and digital signature procedures. Although not as lightweight as the NetCeiver multicast protocol, industry standards like UPnP-AV and the related DVB-IPTV might provide an important basis for interoperability. Thus we plan to integrate the NetCeiver hardware into these frameworks.

REFERENCES

Djama, I. and Ahmed, T. 2006. An mpeg-21-enabled video adaptation engine for universal iptv access. In *in Proc. IEEE Boadband Multimedia Symposium 2006, Las Vegas*.

DVB Consortium. 2009. *DVB-IPTV 1.4: Transport of MPEG 2 TS Based DVB Services over IP Based Networks (and associated XML) (dTS 102 034 V1.4.1)*. http://dvb.org/technology/standards/index.xml#internet.

ETSI. *DVB Standards.* http://www.etsi.org/WebSite/Technologies/DVB.aspx.

Fabio Forno et al. 2006. HoNeY: a MHP-based Platform for HOme NEtwork interoperability. *pp.102-110, 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06)*. http://omero.polito.it/appeal/articoli/Honey.pdf.

IETF. 1998. *Protocol Independent Multicast - Sparse Mode (PIM-SM), RFC2362.* http://tools.ietf.org/html/rfc2362.

IETF. 1999. *Service Location Protocol, Version 2, RFC2608.* http://tools.ietf.org/html/rfc2608.

IETF. 2004. *Multicast Listener Discovery Version 2 (MLDv2) for IPv6, RFC3810.* http://tools.ietf.org/html/rfc3810.

IETF. 2006. *IP Version 6 Addressing Architecture, RFC4291.* http://tools.ietf.org/html/rfc4291.

Ma, H. and Shin, K. G. 2002. Multicast video-on-demand services. *ACM Computer Communication Review 32*, 2002.

RFC Draft. 2008. *Multicast DNS.* http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt.

Sun, H., Vetro, A., Xin, J., Sun, H., Vetro, A., and Xin, J. 2007. An overview of scalable video streaming. *Wireless Communications and Mobile Computing 7*, 159–172.

UPnP Forum. 2008. *UPnP AV Architecture.* http://www.upnp.org/standardizeddcps/default.asp.

W3C. 2004. *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0.* http://www.w3.org/TR/CCPP-struct-vocab/.

Xilinx Inc. *MicroBlaze Processor.* http://www.xilinx.com/products/design_resources/proc_central/microblaze.htm.

Xilinx Inc. *Spartan-3E FPGA Family Data Sheet.* http://www.xilinx.com/support/documentation/spartan-3e_data_sheets.htm.