EVALUATING THE NETWORKING PERFORMANCE OF LINUX-BASED HOME ROUTER PLATFORMS FOR MULTIMEDIA SERVICES

Ingo Kofler, Robert Kuschnig, Hermann Hellwagner

Institute of Information Technology (ITEC)
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria
Email: firstname.lastname@itec.uni-klu.ac.at

ABSTRACT

Wireless router platforms based on the Linux operating system are becoming popular in consumers' home networks. The transmission of multimedia data or their use as media-aware network elements imposes high traffic and computational loads on these devices. Thus, it is interesting to evaluate the networking and processing capabilities of such home router platforms in order to assess their usefulness for improved multimedia services such as in-network H.264/SVC video stream adaptation. This paper presents a performance evaluation of three home router platforms representative for low-end, mid-range, and high-end devices. The scope of the evaluation is the performance of the Linux networking stack on these routers; results for both application-layer (TCP and UDP) transmission and kernel-level (UDP) traffic routing are given. The results show that both TCP and UDP throughputs are significantly below (less than half of) the outgoing (wired) links' nominal capacities and depend very much on the sizes of the transmitted data blocks. This clearly indicates that the networking performance is limited by the platforms' processing capabilities and the lack of mechanisms that offload networking tasks from the CPUs. This behaviour cannot be observed on today's PC systems and has to be considered when deploying multimedia services on these network devices. Furthermore, a detailed analysis of the Linux networking stack reveals that the performance is heavily impacted by the *netfilter* code, even when no packet filtering or network address translation is being performed. Considerable performance gains can be achieved when this *netfilter* code is bypassed.

1. INTRODUCTION

Wireless router platforms based on Linux are becoming more and more popular in the consumers' home networks [1]. Although Linux is not a real-time operating system, it allows to develop cheap network devices with sufficient performance for most home network scenarios. Additionally, using a Linux platform allows manufacturers to offer a rich set of functionality and services that are developed and maintained by the community. Consequently, the platforms can also be used for multimedia services which include the possibility of using such platforms as media-aware network elements (MANE) [2] for in-network adaptation or acting as network attached storage (NAS) to share multimedia content via SMB or UPnP. In our previous work [3] we already investigated the usage of a low-end home router platform for adapting multiple standard-definition H.264/SVC video streams in parallel and demonstrated the feasibility of in-network adaptation on an off-the-shelf router platform. Further evaluations with more recent platforms showed that their performance is sufficient to adapt even multiple high-definition streams.

Most of these services have in common that they are typically implemented on the application layer and run as processes in the user space. This imposes higher requirements on the wireless router platforms which were initially developed and designed for typical networking tasks like routing and bridging. In this work, the processing and networking capabilities of three different residential router platforms are evaluated. The scope of the evaluation is the performance of the networking stack with a strong focus on the application layer performance. The evaluation is not related to the wireless networking performance itself since this depends on many external factors as well as the actual wireless chipset. Instead, we are interested in measuring the performance of the networking stack using the wired network interface. The purpose of this evaluation is to get an in-depth understanding of performance limitations of existing platforms and to identify possible bottlenecks that have to be considered when using them for multimedia services.

2. HARDWARE PLATFORMS

All evaluated router platforms are intended for residential and small office–home office (SOHO) markets and can be considered as representative for millions of legacy devices that are nowadays deployed in home networks. The platforms were selected in a way that both their nominal CPU speed as well as their network connectivity cover a broad range of available

Vendor/Model	Linksys WRT54GL (WRT54)	TP-Link WR1043ND (TLWR)	Ubiquity Routerstation Pro (RSPRO)
SoC	Broadcom BCM5352EL	Atheros AR9132	Atheros AR7161
CPU	MIPS32 4Kc	MIPS 24Kc	MIPS 24Kc
CPU clock	200 MHz	400 MHz	720 MHz
SDRAM size	16 MB	32 MB	128 MB
SDRAM clock	100 MHz	400 MHz	360 MHz
Flash capacity	4 MB	8 MB	16 MB
System bus	SSB	AHB	AHB
Bus speed	100 MHz	200 MHz	180 MHz
Wired	802.3u	802.3z	802.3z
Wireless	802.11b/g	802.11b/g/n	802.11a/b/g/n

Table 1. Overview - Hardware platforms

devices. All have in common that they ship with firmware that is not a dedicated router or real-time operating system but based on a Linux kernel.

The most relevant technical specifications of the evaluated router platforms are given in Table 1. The Linksys WRT54GL wireless router represents the lower end of the performance spectrum. The technical platform is rather old (first released in 2002) and the router is still very popular due to its flexibility and the possibility of replacing the original firmware by a Linux-based third-party firmware. The TP-Link TL-WR1043ND wireless router platform was released in 2008 and also targets the residential and home network market. It is based on an Atheros AR9132 SoC and offers both a fast wired and wireless network interface. Additionally, this platform is equipped with a USB 2.0 host controller that allows to attach different USB devices like mass storage devices for network sharing. The Ubiquity Networks Routerstation Pro marks the top end of wireless router platforms under investigation. It is based on the Atheros AR7161 SoC, which is designed as a high performance wireless network processor that enables efficient designs for triple-play services. All of the platforms are based on MIPS CPUs which are running at clock rates ranging from 200 to 720 MHz. This range of the clock rates can be considered as representative for the wireless router platforms available at the time of writing. As an aside, the same series of SoCs (AR9132 and AR7161) are also used in wireless routers of other well-known vendors like D-Link, Netgear, Mikrotik, or Buffalo. In order to have a common software basis for evaluation, the original vendor firmware was replaced by OpenWrt[1]. OpenWrt is a Linux-based firmware which is open-source and can be used on a broad range of embedded platforms. For the evaluations, the OpenWrt release 10.03 (codename Backfire) with Linux kernel 2.6.33 was used as a common basis.

[1]http://www.openwrt.org

Platform	WRT54	TLWR	RSPRO	Core i7
CoreMark	332	701	1263	9457
CPU clock rate [MHz]	200	400	720	2670
CoreMark/MHz	1.66	1.75	1.75	3.54
STREAM copy [MB/s]	59	231	381	6830

Table 2. Benchmark results

3. BASIC PLATFORM PERFORMANCE

In a first step, the processor and memory performance of the three router platforms were evaluated. The platforms were assessed by means of two different benchmarks to get some insight into the computational performance that one can expect from these platforms and to compare their performance with a modern desktop system. The CoreMark benchmark (version 1.0)[2] was used to assess the performance of the CPU core of the router platforms. For comparisons, it is recommended to normalize the benchmark score by dividing it by the CPU clock rate. In addition to the CPU performance, also the performance of the memory system was evaluated. For that purpose, the STREAM benchmark was used to determine the throughput of memory copy operations.

The results of both benchmarks are summarized in Table 2. For a comparison of the computing performance, the benchmark was also executed on a modern PC system. It should be noted that the benchmark was executed single-threaded, so only the performance of a single core was assessed. As one can see from the CoreMark results, the performance of the router platforms highly correlates with the clock rate of the platforms which gets even more obvious when comparing the normalized CoreMark values. The TLWR and RSPRO score identically at 1.75 CoreMark/MHz since both are based on the same MIPS32 24Kc CPU. When comparing the absolute CoreMark values of the embedded router platforms with the Intel Core i7 M620 processor, one can conclude that just one of the latter processor's cores is between 7.5 times (RSPRO) and 28 times (WRT54) faster. The results of the STREAM copy benchmark indicate that the memory performance of the WRT54 platform is rather modest and considerably lower than that of the TLWR and RSPRO platforms. Some of the reasons are the low SDRAM clock rate (100 MHz) of the WRT54 and its smaller cache lines that cause a higher miss rate for the same sequential memory access pattern. Not surprisingly, the score obtained for the STREAM copy benchmark[3] on the PC system is much higher. The memory throughput of the embedded platforms is only between $1/18$ (RSPRO) and $1/114$ (WRT54) of that of the Core i7 system. As discussed later, this memory bottleneck also negatively influences the networking performance of the platforms.

[2]http://www.coremark.org

[3]http://www.cs.virginia.edu/stream/

4. EVALUATION SETUP

In a next step, we evaluated the wired networking performance in terms of achievable throughput on the three platforms. The testbed consists of two PCs that are connected to the router. One computer (PC A) is connected with the WAN port of the router, while the second one (PC B) is connected to one of the LAN ports. Both PCs are equipped with Gigabit NICs and are connected to the router using appropriate cables. Each of the two PCs runs the Ubuntu 9.10 operating system with Linux kernel version 2.6.31.

The throughput measurement was performed by using the command line tools iperf[4] and netperf[5]. The *iperf* tool can be used to measure the throughput of both UDP and TCP traffic between two systems. The tool allows amongst others to specify certain parameters like the length of the read/write block size for the socket API calls as well as different socket buffer sizes. In case of UDP traffic, it is also possible to specify the transmission rate according to which the UDP traffic is generated. In the course of the experiments, it turned out that it was not possible to force iperf to send UDP packets at maximum rate. Even setting very high values for the transmission rate led to inferior bandwidth utilization. Therefore, the *netperf* tool was used in cases where it was required to generate a UDP stream at maximum rate. All the experiments were performed with a duration of 20 seconds each and were repeated five times; arithmetic means were then calculated.

The following three traffic scenarios were used for the evaluation:

Outgoing TCP. In this first setting the achievable TCP throughput from the router to PC B was measured using iperf. The term "outgoing" is considered as relative to the router, which means that the router is the traffic source while PC B acts as the sink. It should be noted that the purpose of this scenario is to evaluate the throughput that a user space process, e.g., a UPnP server or a proxy, can expect on such a platform. Different values for the read/write block size were selected for evaluation.

Outgoing UDP. In addition to TCP, also the use of the simpler UDP protocol was investigated by using netperf. Again, "outgoing" is related to the router which means that the UDP traffic was generated at the router and sent to PC B. In contrast to the first scenario, the achievable UDP throughput is relevant in the context of UDP-based services, e.g., for an RTP-based adaptation proxy as described in [3]. As with the outgoing TCP traffic the read/write block size was varied to investigate its impact on the throughput. As the block size influences the packet size, the range was chosen by considering the MTU size of the Ethernet link as the maximum block size, in order to avoid fragmentation on the IP layer.

[4]http://iperf.sourceforge.net

[5]http://www.netperf.org

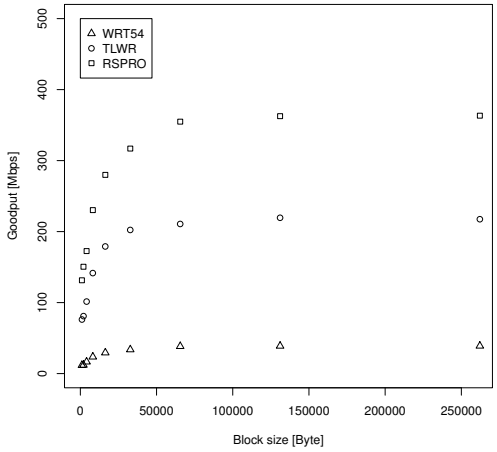

Fig. 1. Throughput – Outgoing TCP traffic

Routing UDP. The last traffic scenario that was used for evaluation is the routing of UDP traffic. In contrast to the traffic scenarios described before, the router neither acts as a traffic source nor as a traffic sink. In fact, the UDP traffic at different transmission rates is generated by PC A and routed through the router to PC B. The important difference to all other scenarios is that routing solely takes place in kernel space. The networking performance at this layer can be considered as relevant for layered multicast approaches as indicated in [2]. The achieved routing throughput for different transmission rates at PC A is measured at PC B by using iperf. As the routing is done on the IP layer, the handling is the same for any other transport protocol like TCP or DCCP. We decided to evaluate the routing performance using UDP since its lack of a flow control mechanism allows us to gradually increase the transmit rate at PC A.

In the course of our experiments we also investigated scenarios with both incoming TCP and UDP traffic. However, due to space restrictions the results cannot be discussed in this paper.

5. EVALUATION RESULTS

For the first evaluation, the outgoing TCP bandwidth at the router was measured by using the iperf tool. It soon turned out that setting the socket *buffer* size to different values had no impact on the achieved throughput. Therefore, a manual setting of this value was no longer used for the rest of the evaluation. Omitting a manual setting forces the kernel to perform an automatic tuning of the buffer.

The results of the outgoing TCP bandwidth measurements are shown in Figure 1. The curves show that the TCP throughput from the router to another device is far below the capacity of the link which is nominally 1 Gbps for the RSPRO and TLWR platforms and 100 Mbps for the WRT54. Another interesting observation is that on these platforms the *block* size

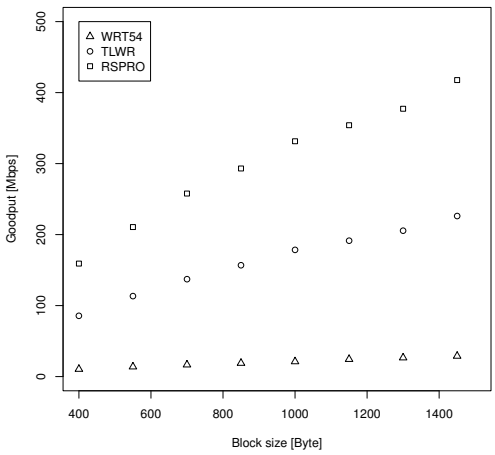

Fig. 2. Throughput – Outgoing UDP traffic

has a significant influence on the achieved throughput. For the RSPRO platform the achieved TCP throughput converges to the maximum of about 363 Mbps only when using block sizes of at least 128 KiB. When using the default block size that is used by iperf (8 KiB), the throughput is only 230 Mbps which is less than $2/3$ of the maximum. The similar drastic behaviour can be observed for the TLWR platform which reaches a maximum outgoing TCP throughput of 217 Mbps for block sizes of 128 KiB and larger. When using smaller block sizes of 8 and 1 KiB the achieved throughput drops to 142 Mbps (65 percent) and 76 Mbps (35 percent), respectively. The WRT54 platform shows a similar behaviour as the RSPRO and TLWR platforms. The maximum throughput of 39 Mbps is only achieved when using large block sizes; selecting a block size of 8 KiB leads to a throughput of only 61 percent of the maximum. One conclusion from this observation is to avoid using small block sizes for transmitting via TCP connections. The overhead of the socket API call and the system call appears to be high on such embedded platforms and only amortizes by using large block sizes. This behavior cannot be observed when measuring the TCP throughput on modern PC-based platforms. For the sake of reference, both PCs were connected directly by an Ethernet cable and the same measurement procedure was applied. Since the network interface of the PC also supports different offload capabilities, the measurements were performed with and without any offloading mechanism. The results showed that the achieved throughput was about 930 Mbps and independent of the chosen block size and offloading. The only observable difference was the different levels of CPU utilization on the PC which reached up to 30 percent when disabling offloading. From these observations one can conclude that on the embedded platforms the achieved TCP throughput is clearly limited by the platform's processing power in combination with the lack of offloading mechanisms.

The results of the outgoing UDP throughput measurements are presented in Figure 2. As one can learn from the plot, there is an almost linear relationship between the block size (i.e., the UDP payload size) and the achieved UDP throughput. Since the achieved throughput is less than half of the link's nominal capacity even in the case of the best-performing platform, it is obvious that the throughput is limited by the processing capabilities of the router. When using a block size of 1450 Bytes, the achieved outgoing UDP throughput is 418 Mbps for the RSPRO platform and 226 Mbps for the TLWR platform. The WRT54 platform achieves up to 29 Mbps of outgoing UDP traffic. The previous workloads have in common that they measure the throughput at the application layer. The last scenario simply measures the routing performance of UDP traffic without any interaction with the user space. In this set-up, PC A generates UDP traffic at different transmission rates. The UDP packets are routed by the router platform to PC B, which acts as traffic sink and as location for the throughput measurement. The transmission rate at PC A was steadily increased in steps of 5 Mbps and the achieved throughput at PC B for each transmission rate was determined. For this evaluation all firewall rules that might block incoming traffic from the WAN port or perform network address translation (NAT) were removed. The direction of the UDP traffic from the WAN to the LAN was selected as this reflects the main direction of traffic in video streaming scenarios. Nevertheless, the tests were also performed in the opposite direction and it turned out that the direction does not have an impact on the routing performance. The results of the routing UDP throughput evaluation are visualized in Figure 3. The curves show a similar behavior for all of the three platforms. Up to a certain transmission rate, the router is capable of routing the traffic. At a certain point, the achieved throughput decreases due to excessive load on the router. The best performance can be achieved by the RSPRO platform which can route up to 430 Mbps of UDP traffic without causing packet loss. On the TLWR platform the maximum UDP throughput can be achieved at a transmission rate of 270 Mbps. The WRT54 platform achieves 70 Mbps for routing the UDP traffic.

6. ANALYZING THE LINUX NETWORKING STACK

As the results in the previous section show, the networking performance highly depends on the block size, especially for the UDP transmission. Since UDP is often used for video streaming applications, an in-depth analysis of the UDP implementation in the networking stack was conducted. Besides, the the protocol implementation is less complex compared to TCP, which makes it easier to trace and identify the performance bottleneck of the platforms. For analyzing the execution of the networking code, the *ftrace* framework [4] of the Linux kernel was used. The purpose of this first analysis was to identify which parts of the kernel source code were executed as a result of a system call. The analysis showed

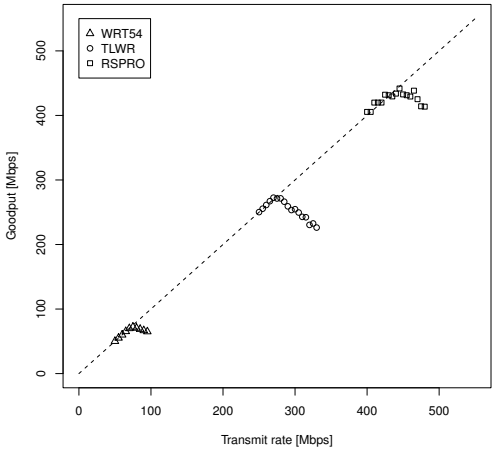

Fig. 3. Throughput – Routing UDP traffic

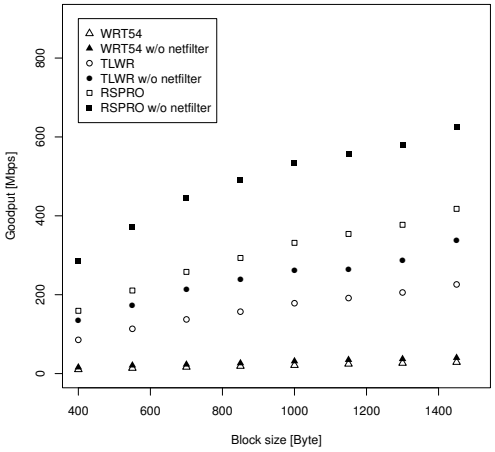

Fig. 4. Outgoing UDP traffic without netfilter

that during a system call for sending a single UDP datagram, about 200 function calls in the kernel take place. Surprisingly, nearly half of the executed functions are related to the *netfilter* code. The netfilter framework in the networking stack of the Linux kernel is used for packet filtering, network address translation and other packet manipulation purposes. As indicated by the analysis, a significant amount of function calls is related to that kernel component, even when no firewall or NAT rules are used on the system. In order to quantify the impact of the netfilter component on the networking performance, the same measurements as in Section 5 were performed *without* the netfilter framework. For that purpose, firmware images for the three hardware platforms were compiled with a kernel configuration with disabled netfilter support. As both the hardware platform and the rest of the software stack remained the same, the influence of the netfilter framework can be exactly investigated. As one can learn from Figure 4, the achieved outgoing UDP throughput significantly increases when using a firmware image without built-in netfilter code. This observation is most obvious for small packet sizes of 400 Bytes where the per-packet overhead of the netfilter code has the highest impact. For such small packet sizes, the performance increase is between 53 percent on the WRT54 platform and up to 80 percent on the RSPRO platform. Obviously, the relative performance increase gets lower when using larger packet sizes but remains at a high level. When choosing packet sizes near the MTU of the Ethernet link, the performance gain is still between 39 (WRT54) and 50 percent (RSPRO). Similar to the outgoing UDP traffic performance, the other traffic scenarios benefit from removing the netfilter code. Due to space restrictions, the plots cannot be shown in this paper but the results can be summarized as follows. The results for the outgoing TCP throughput show that the throughput obtained with the firmware images without the netfilter code are significantly higher. The amount of improvement depends on the block size which is obvious since most of the function calls to the netfilter code are done once per system call. Obviously, the relative increase of TCP throughput is higher for small block sizes (about 1 to 4 KiB) than for larger ones ($\geq$ 16 KiB). On the RSPRO platform, increases of 90 percent can be observed for small block sizes. Although the relative improvement deteriorates with increasing block sizes, the improvement is still 28 percent for a block size of 16 KiB. A similar behavior can be observed for both the TLWR and WRT54 platforms. On the TLWR platform, the improvements range from 89 percent for small block sizes to 9 percent for larger ones. The maximum improvement on the WRT54 platform is about 38 percent which reduces to 15 percent when increasing the block size. The UDP routing performance benefits from removing the netfilter code as follows. On the RSPRO platform, the routing throughput is increased by 170 Mbps to 600 Mbps (40 percent). Comparable performance improvements are also evident in the case of the other two platforms. Without the netfilter code, the TLWR platform achieves a throughput of 355 Mbps which is a plus of 85 Mbps or 31 percent. On the WRT54 platform, the routing throughput increases to 95 Mbps (plus 25 Mbps or 36 percent).

Based on the Linux kernel image with a disabled netfilter component, the ftrace framework was further used to identify where the execution time in the networking code is spent. The evaluation was performed on the RSPRO platform for packet sizes of 400 and 1450 Bytes. The contribution of each building block in the networking stack to the total execution time was determined using the ftrace framework. As shown in Figure 5, most of the clock cycles are spent for buffer handling. This block includes allocating a socket buffer structure in the kernel space, copying the data from the user space, and calculating the checksum. The overhead caused for executing the code responsible for the socket API, the IP layer (IP header, routing, neighbouring protocol) and UDP layer is quite independent of the packet size. Also the time spent

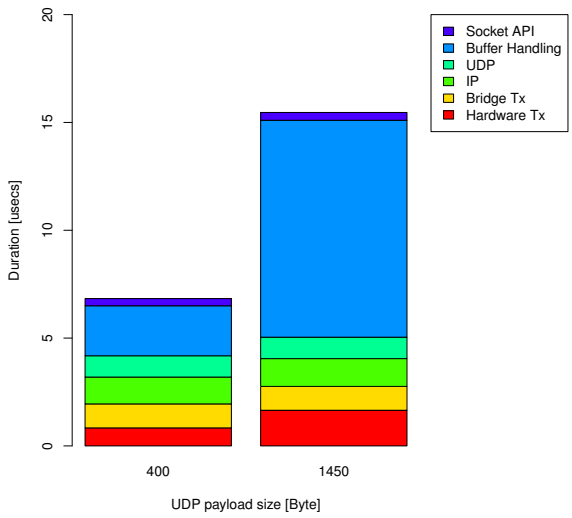

Fig. 5. Execution time - UDP sendto syscall without netfilter

in the code for realizing the L2 bridging between the LAN and wireless interface (Bridge Tx) is constant. Finally, the execution of the driver code (Hardware Tx) depends on the packet size again. Since the network interface uses DMA, the driver has to ensure consistency between the CPU cache and the main memory before starting the DMA operation. The time for transfering the packet's data from the cache to the main memory obviously depends on the size of the packet again. A consequence of the amount of fixed overhead per system call is the significant impact of the block size on the achieved throughput. This is consistent with the results discussed in Section 5. When considering packet sizes close to the MTU of the Ethernet link, the memory bandwidth of the platform and the lack of offloading capabilities for checksum calculation are the limiting factors.

7. RELATED WORK

To the best of our knowledge this work is the first that investigates the performance and implications of the Linux networking stack on home router platforms. The need for investigating these aspects is grounded in our work on in-network adaptation of H.264/SVC content on such devices [3] as it is performed on the application layer by a user space process. As more and more devices in this market segment are based on Linux the insights obtained can be applied to a variety of different networking devices. Another evaluation of home router platforms is given in [5], however, the focus of this work is more related to NAT and its implementation on the devices. The work presented in this paper, however, focuses on the throughput that can be achieved by (multimedia) services running in user space on the router platform. Another currently ongoing work is the proposed home networking working group within the IETF which targets the role of such router platforms in home networks.

8. CONCLUSIONS

This paper provides a performance evaluation of Linux-based home router platforms which focuses on aspects relevant for multimedia services. The evaluation of the home routers' networking performance disclosed that the TCP and UDP throughputs achievable by user space processes on these devices are far less than the nominal link speeds. The limiting factors are moderate processing power and the lack of offloading capabilities. Another crucial aspect is the selection of the block size used during socket API calls since it highly influences the achieved throughput. This factor does not play a role on modern server or desktop systems, yet should be taken into account for good performance of multimedia services delivered via home routers. E.g., in the case of TCP traffic it is recommendable to use block sizes of 32 KiB or even larger to obtain a reasonable throughput. A detailed execution trace of the Linux networking stack further shows that the *netfilter* framework in the Linux networking stack has significant influence on the throughput. The removal of this component, if possible in a given application scenario, leads to a throughput increase of around 50 percent and should be considered as a viable option. The netfilter component can be omitted if the router is used internally only, e.g., serving as wireless access point and serving multimedia content via UPnP or SMB. As the platforms investigated in the paper cover the performance spectrum of typical home router platforms, the performance figures provided can be considered as representative.

9. REFERENCES

[1] Alexander Sirotkin, "Building a next-generation residential gateway," *Linux Journal*, vol. 2007, no. 160, pp. 5, 2007.

[2] Stephan Wenger, Ye-Kui Wang, and Thomas Schierl, "Transport and Signaling of SVC in IP Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1164–1173, Sept. 2007.

[3] Ingo Kofler, Martin Prangl, Robert Kuschnig, and Hermann Hellwagner, "An H.264/SVC-based adaptation proxy on a WiFi router," in *Proceedings of the NOSSDAV'08*, May 2008, pp. 63–68.

[4] Tim Bird, "Measuring function duration with ftrace," in *Proceedings of the Linux Symposion 2009*, July 2009, pp. 47–54.

[5] Seppo Hätönen, Aki Nyrhinen, Lars Eggert, Stephen Strowes, Pasi Sarolahti, and Markku Kojo, "An experimental study of home gateway characteristics," in *Proceedings of the ICM'10*. 2010, pp. 260–266, ACM.