Reduced Switching Delay for Networked Music Performance

George Xylomenos, Christos Tsilopoulos, Yannis Thomas and George C. Polyzos
Mobile Multimedia Laboratory, Department of Informatics
Athens University of Economics and Business
Athens 10434, Greece
{xgeorge,tsilochr,thomasi,polyzos}@aueb.gr

***Abstract*—Networked Music Performance (NMP), where musicians located in different places perform together in real time via a network, requires extremely low end-to-end delay. As part of an effort to reduce all aspects of NMP delay, we are focusing on the packet processing and replication delay at the centralized node which enables communication between the multiple NMP participants. We propose three approaches to dramatically reduce this delay, based on both hardware and software optimizations.**

***Index Terms*—Networked music performance, multipoint conferencing unit, delay**

I. Introduction

Among real-time multimedia applications, *Networked Music Performance* (NMP), where musicians located at different places perform together via a network, has probably the most stringent delay requirements, as proper interaction between musicians has a very low delay tolerance. Considering NMP as a special case of conferencing applications, there are many obvious sources of delay to attack, including media capture, coding and packetization, as well as network transmission and queuing. A less obvious target is the delay introduced by the *Multipoint Conferencing Unit* (MCU), the centralized server which relays data between all NMP participants. MCUs are normally application layer processes which receive media packets from each participant, decide where to forward each packet, replicate it accordingly and inject the copies to the network. Since packet copying between the kernel and applications is a notorious source of delay, we propose three different strategies for reducing this delay, as part of a comprehensive project on advancing the state of the art in NMP.

II. The MusiNet approach

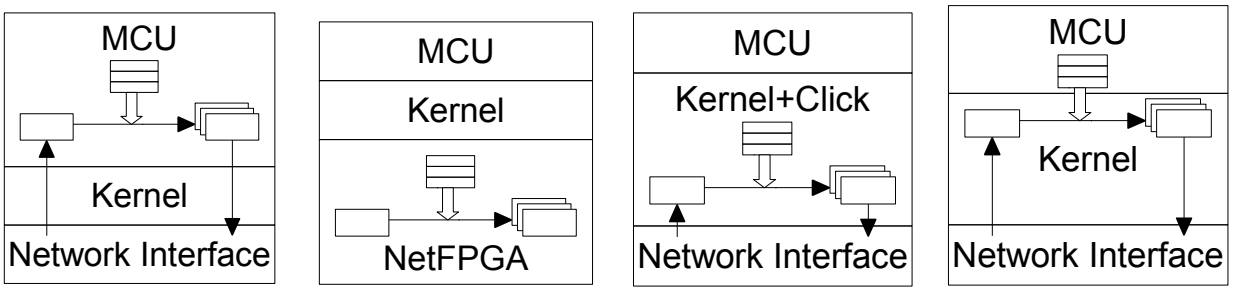
The MusiNet project [1], a collaboration among groups at the University of Crete, the University of Athens, the Technological Institute of Crete, and the Athens University of Economics and Business, is building a system that will advance the NMP state of the art in many fronts. In an NMP context, the acceptable upper bound of end-to-end latency has been reported to be as low as 25 ms [2], compared to the 150 ms considered acceptable for audio conferencing, therefore our primary target is reducing delay in all stages of an NMP session. The main tasks of the project focus on ultra-low delay audio and video coding, based on recent advances in these fields, coupled with optimizing the media capture and packetization procedures. On the other hand, the transmission and queueing delays between the communicating endpoints are beyond our scope; when using the Internet for communication, the best we can do is attempt to use lightly loaded paths.

There is, however, another element of delay between the endpoints: the MCU which receives media from all NMP participants and relays them appropriately. A traditional MCU decodes all incoming media streams, composes them (e.g. mixing all audio streams or selecting only the current speaker, and composing all video streams into a single video pane), re-encodes the result and separately transmits it to each participant. In an NMP system participants would prefer to choose how to mix the media streams themselves, e.g. the drummer of a rock band may want to hear the bass guitar more than the vocals. For this reason, in the MusiNet project the MCU is a media router rather than a media transcoder [3]: each participant indicates to the MCU which media streams it desires to receive, and the MCU simply replicates and forwards these streams, without processing them.

Even this approach introduces MCU delays of up to 20 ms [4], which, while perfectly acceptable for conferencing, are disastrous for NMP. Therefore, some NMP approaches employ direct communication between participants [5]. This is inefficient with multiple participants, as it requires transmitting a copy of each media stream to every other participant. For this reason, in previous work we studied the option of avoiding the MCU by letting NMP participants directly multicast their media streams to all other participants [6]. While this does indeed reduce latency, it requires network protocols that are not currently deployed on the Internet at large. As a result, the MCU remains essential for media routing, hence it makes sense to reduce its latency as much as possible.

III. Building an ultra-low delay MCU

In order for an MCU to properly perform its packet routing role, it receives signaling packets and media packets from the NMP participants. The signaling packets indicate which media streams are desired by a recipient, i.e. which video and audio streams the participant wants to receive. The media packets contain the media streams produced by each participant. Based on the signaling packets and, possibly, on prevailing network conditions, the MCU maintains a table (or other data structure) indicating how to treat the packets of each media stream, i.e.



(a) (b) (c) (d)

Fig. 1. (a) Regular MCU operation, (b) MCU with NetFPGA, (c) MCU with Click, (d) MCU with netmap.

drop them, or replicate them for each participant that has requested them. The data flow in the MCU is therefore as depicted in Figure 1(a): packets are passed to the MCU process by the kernel, the MCU replicates them for each recipient, and the replicated packets are passed to the kernel for transmission. This requires context switching between the kernel and the application, as well as data copying which grows with the number of participants. Both these activities are well-known sources of delay [7]. Considering the simple task that the MCU performs in our context, it is only natural to look for options to avoid this delay; we are considering the following three.

The first option is to ask the hardware for help. While a standard network interface cannot perform complex table lookups, this is possible with the NetFPGA boards, where a set of network interfaces are coupled with a Field Programmable Gate Array that can be programmed to perform arbitrary tasks [8]. A possible MCU architecture exploiting a NetFPGA is shown in Figure 1(b). The application handles the signaling packets, which are not time critical, while the NetFPGA handles the time critical tasks of receiving, copying and transmitting the media packets. A table indicating how to treat each packet resides in the NetFPGA memory; this table is modified based on the signaling packets received by the application. As a result, the entire routing procedure executes in the hardware, not even interrupting the kernel. Furthermore, the NetFPGA can avoid copying a packet multiple times: it can instead rewrite its header and transmit the same packet multiple times, thus reducing memory bandwidth requirements.

The second option is to use the Click modular router [9], a toolkit for composing software routers from a set of simpler modules. Click can run either at the user or at the kernel level, with the former option being more useful for testing and debugging, and the latter being more useful for optimized performance. As in the previous case, the router can be split in two parts, a control part residing at the application level, and a routing part residing at the kernel level, as shown in Figure 1(c). Compared to the NetFPGA option, Click is easier to program (in C++ rather than in VHDL), it allows testing to take place at the user level, but, while it avoids most packet copying and context switching due to in-kernel execution, it takes up processor time and may not be able to perform optimizations like packet replication without copying.

The third option is to use the netmap framework for packet handling at the application level [7]. In this case, the entire MCU resides at the user level, as shown in Figure 1(d), so while context switching does take place, no packet exchanges are needed between the kernel and the user level, since the MCU can directly manipulate the media packets in kernel memory. Previous work has shown that with the netmap framework user-level Click routers can match the performance of their kernel-level counterparts [7], therefore an MCU written directly over netmap should be able to at least match the performance of a kernel-level MCU written in Click.

IV. Conclusion and Future Work

We presented three options for dramatically reducing the latency of an MCU by avoiding context switching and excessive packet copying between the application and the kernel. We are currently working on both the Click and netmap approaches. Since the NetFPGA approach requires specialized hardware and is the most complex to program, it will only be pursued if the other two approaches do not yield sufficient latency gains.

Acknowledgment

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: THALIS - University of Crete - MUSINET.

References

[1] The MusiNet project. [Online]. Available: http://musinet.aueb.gr/
[2] A. Renaud, A. Carôt, and P. Rebelo, "Networked music performance: State of the art," in *Proc. of the AES International Conference*, 2007.
[3] A. Eleftheriadis, R. M. Civanlar, and O. Shapiro, "Multipoint videoconferencing with scalable video coding," *Journal of Shejiang University SCIENCE A*, vol. 7, pp. 696–705, 2006.
[4] VidyoRouter datasheet. [Online]. Available: http://www.vidyo.com/wp-content/uploads/DS-VidyoRouter.pdf
[5] Soundjack. [Online]. Available: http://www.carot.de/soundjack/
[6] C. Stais, Y. Thomas, G. Xylomenos, and C. Tsilopoulos, "Networked music performance over information-centric networks," in *Proc. of the IEEE IIMC*, 2013.
[7] L. Rizzo, "Netmap: a novel framework for fast packet i/o," in *Proc. of the USENIX ATC*, 2012.
[8] J. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "NetFPGA–an open platform for gigabit-rate network switching and routing," in *Proc. of the IEEE MSE*, 2007.
[9] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, pp. 263–297, 2000.