xStreamer: Modular Multimedia Streaming

Alexis Rombaut Nicolas Staelens Nick Vercammen

Brecht Vermeulen Piet Demeester

Ghent University - IBBT, Department of Information Technology (INTEC)
G. Crommenlaan 8, Bus 201
B-9050 Ghent, Belgium
+32 9 331 49 79
{alexis.rombaut, nicolas.staelens, nick.vercammen, brecht.vermeulen, piet.demeester}@intec.ugent.be

Categories and Subject Descriptors: D.0 [Software]: GENERAL

General Terms: Design, Experimentation

Keywords: Streaming, Modular, Multmedia, Open Source

1. INTRODUCTION

The xStreamer intends to be a flexible and modular open source streamer. The selection of current open source streamers which support both video and audio is limited, with VLC Media Player [1], Darwin Streaming Server [2] and Helix DNA Server [5] being the foremost solutions. The xStreamer distinguishes itself by providing a modularity that goes beyond the mere modular programming offered by the current open source solutions and that manifests itself in how the user controls and configures the streamer.

The distribution of the program is available at [6], a newly launched web site. How to the build the program is on various platforms is described in *INSTALL.txt*. The document (*doc/demo.pdf*) supplements this short introduction to the xStreamer by demonstrating the applications of the program in five demos. Furthermore, the distribution contains a detailed practical manual (*doc/manual.pdf*) on how to build configurations together with a complete overview of every component and its parameters.

1.1 Modularity

The modularity is inspired by the Click Modular Router project [4] and operates by offering components which perform basic functions such as reading video frames from a file, classifying packets based on their frame type or randomly discarding packets with a given probability, etc. The user builds the streamer by combining a collection of these components in a directed graph: the vertices form the components and the directed edges form flows of packets from one component to another. Figure 1a shows as an example the graph of a simple streaming solution: a reader parsing a video file and outputting one packet per frame, a packetizer splitting each packet into a series of packets, each smaller than for example 1500 bytes, a scheduler releasing each packet at the instant corresponding with the timestamp of the packet and finally, a transmitter adding network headers and sending the packet over the network. By means of simple changes to the graph, the streamer performs additional or similar functions, providing a part of the flexibility of the xStreamer. For example, the streamer in figure 1b performs a similar function to figure 1a, but instead of sending the packets over a single connection, the component *classifier* splits the flow of packets into three flows based on for example the frame type (I, P or B), with each flow sent over a separate network connection.

Additionally, the xStreamer has components performing the reverse operations of streaming, such as receiving, unpacketizing and writing, allowing it to offer a proxy function which redirects for example the three connections from figure 1b into a single new connection or a capture function which redirects the received packets to a file. Figure 2 shows a configuration performing the reverse operation from figure 1b and providing a proxy as well as a capture function.

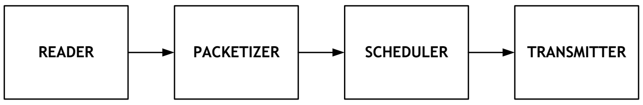
Since the scheduler component introduces real time behaviour to the passing flow of packets, without such a component, the program can function as a video tool, working offline, for example in order to impair a video bit stream with a given packet loss rate as shown in figure 3. The program works as fast as possible in this case, processing a stream in a fraction of the actual duration of the bit stream. Video tool possibilities include controlled or random impairment, elementary stream extraction, statistics (average bit rate, packet size, etc.), plots of the bit rate evolution and transcoding (experimental but future work will improve this).

In addition, the scheduler component can introduce simulated time behaviour, allowing the streamer to function as simulator where time increases with arbitrary increments of for example 1 ms. This choice between real time, simulation and offline function, provides the xStreamer with further flexibility.

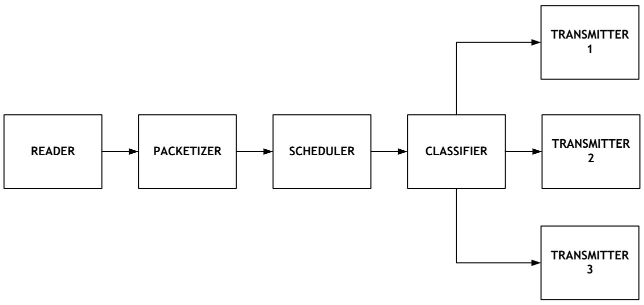
2. FEATURES

The supported codecs, containers and network protocols form a crucial aspect of a streamer. The program can handle any container which the underlying library *avformat* [3] supports. Nevertheless, the program cannot packetize the extracted streams from these containers if the corresponding packetizer is not supported. The internal generic packetizer of the xStreamer can handle such streams but this pack-

MM'09, October 19–24, 2009, Beijing, China.
ACM 978-1-60558-608-3/09/10.



(a) Basic streamer



(b) Differentiated streamer

Figure 1: Directed graph configuring the xStreamer, showing in (a) a basic streamer and in (b) a differentiated streamer where the component *classifier* splits the video stream into three based on for example the frame type.

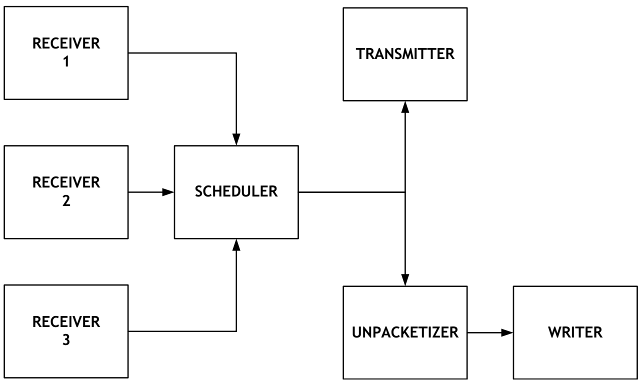


Figure 2: Directed graph configuring the xStreamer to rejoin the stream from figure 1b, sent over three connections, into one and afterwards sending it over a single connection (proxy function) as well as writing it to a file (capture function).

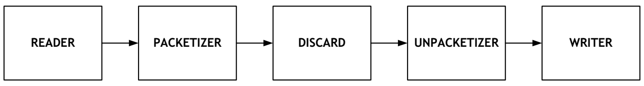


Figure 3: Directed graph configuring the xStreamer as impairment tool which drops parts of frames. The component *discard* drops the packets it receives with a given probability.

etization is not supported by other programs and players. However, a standardized packetizer is available for any of the codecs listed below. The program supports RTP (Realtime Transport Protocol), an important protocol for streaming, together with RTSP (Real Time Streaming Protocol) which manages a collection of RTP connections. In addition, the program supports the standard transport protocols UDP and TCP. The manual provides a detailed overview of the supported codecs and containers.

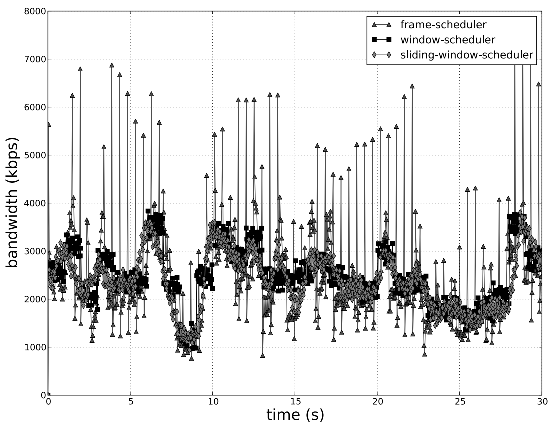


Figure 4: Bit rate evolution: (1) scheduling the packets per frame, (2) scheduling the packets over a fixed window of 1000 ms and (3) scheduling the packets over a sliding window of 1000 ms.

Features of the xStreamer include modularity, flexibility and differentiated streaming as demonstrated in the introduction. Furthermore, the program offers SVC (Scalable Video Coding) support, which distinguishes it from the current open source solutions. Additionally, the xStreamer has a wide selection of schedulers, each offering a different smoothing of the bit rate. These schedulers offer for example smoothing per frame, GOP, fixed window, etc. Figure 4 shows the effect of smoothing per frame, over a fixed window (1000 ms) and over a sliding window (1000 ms) respectively for the same sequence by plotting the instant bit rate at 40 ms intervals.

3. CONCLUSION

The public release of the xStreamer, under the open source license GPL (GNU Public License), coincides with the submission to the Open Source Software Competition. The xStreamer distinguishes itself from current solutions by offering a novel configuration method based on graphs of components performing elementary video functions.

4. REFERENCES

[1] L. Aimar, G. Bazin, et al. VLC media player. `http://www.videolan.org/vlc/`, 2009.

[2] Apple Inc. Darwin Streaming Server. `http://dss.macosforge.org/`, 2008.

[3] F. Bellard, M. Niedermayer, et al. FFmpeg. `http://www.ffmpeg.org/`, 2009.

[4] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The Click modular router. *SIGOPS Oper. Syst. Rev.*, 33(5):217–231, 1999.

[5] RealNetworks. The Helix DNA Server. `https://helix-server.helixcommunity.org/`, 2008.

[6] A. Rombaut. xStreamer: Modular Multimedia Streaming. `http://xstreamer.atlantis.ugent.be`, 2009.