MCN Streaming

An Adaptive Video Streaming Platform *

Qin Chen
Advisor: Prof. Dapeng Oliver Wu

Multimedia Communications and Networking (MCN) Lab
Dept. of Electrical & Computer Engineering, University of Florida,
Gainesville, FL 32611, USA

Contents

*Last updated on May 1, 2010. Email: eric.qin.chen@gmail.com

1 Introduction

Video streaming has gained its popularity in both academia and industry. With the prevalence of mobile computing, it poses new challenges and opportunities. A video encoder that adapts to the instant channel conditions has the potential to provide better video quality even under hostile transmission environments. The motivation of MCN Streaming is to provide a complete video streaming platform that can ease the experimenting of various adaptive video encoding algorithms over the real-world networks, instead of merely depending on simulations.

To summarize, MCN Streaming is:

- a complete video streaming software, including the encoder/sender and receiver/decoder.
- a platform for experimenting adaptive video coding algorithms.
- flexible and extensible.

2 Architecture

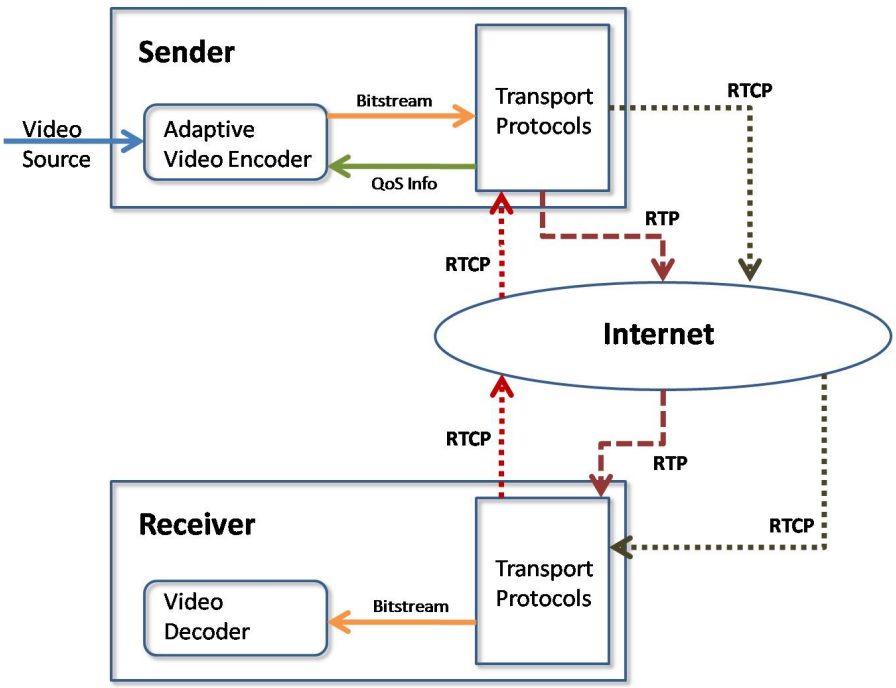

Figure 1: Overview of MCN Streaming architecture.

Fig. 1 shows the high level overview of the MCN Streaming architecture. MCN Streaming consists of two components: a sender and a receiver. Sender is responsible for video encoding and transport layer packet encoding, while receiver is responsible for transport layer packet decoding and video decoding. Encoded packets travel through the Internet. Receiver also sends feedback to receiver via RTCP packets. The feedback (QoS information) is used by video encoder to

adjust its encoding parameters on-the-fly. By doing this, video encoder adapts to the instant channel conditions and better video quality is expected at the receiver side [1].

3 Using MCN Streaming

3.1 Dependencies

MCN Streaming is in fact a GStreamer application. Thus, in order to build and use MCN streaming, you need to have GStreamer and its modules installed first.

Most, if not all, Linux distributions provide packages of GStreamer. You should find these in your distribution's package repository. Below is a list of all GStreamer modules that are necessary to run MCN Streaming.

- GStreamer: core library and elements
- gst-plugins-base: an essential exemplary set of elements
- gst-plugins: additional elements
- gst-ffmpeg: FFmpeg-based plug-in
- gst-plugins-good: a set of good-quality plug-ins under LGPL license
- gst-plugins-bad: a set of plug-ins that need more quality
- gst-plugins-ugly: a set of good-quality plug-ins that might pose distribution problems

If any of the components is not included in your Linux distribution, you can always download the source codes and build it.

3.2 Build and Install

MCN Streaming software package uses GNU Autoconf and Automake tools. Once you download the source codes, you can build and install MCN Streaming in the usual "**./configure**", "**make**", and "**sudo make install**" way.

Once you successfully build and install MCN streaming, you shall get two binaries: **mcn_sender** and **mcn_receiver**, which is the video encoder/sender and receiver/decoder, respectively.

MCN Streaming is very flexible in terms of selecting encoding and transmitting parameters. These parameters can be specified from command line options, in case you wish to override the default values. To get a complete list of the sender options, type "**mcn_sender --help-sender**" in the terminal. To get a complete list of the receiver options, type "**mcn_receiver --help-receiver**" in the terminal. We will cover these options in more details later.

3.3 Sender and Receiver Examples

Here we give simple examples that show how to use MCN Streaming to send and receive real-time video.

At one terminal, type following command.

```
mcn_sender -s videotestsrc -e x264enc -f 20 -w 352 -h 288 -b 300
```

[1] In current version of MCN Streaming, the adaptive video encoding algorithm has not been implemented yet, i.e., RTCP packets are parsed but video encoder does not actually take advantage of the information.

This command indicates to encode a GStreamer test video source into an H.264 bitstream at a frame rate of 20. The video resolution is 352×288 and the bit-rate is 300kbps. The encoded raw video bitstream will be further encoded into corresponding RTP packets and send to the receiver. Note that RTCP packets are also sent to receiver.

The default receiver is localhost. So we can type following command at another terminal to receive the video.

```
mcn_receiver -d ffdec_h264
```

The "-d" option tells the receiver to use an H.264 decoder to decode the incoming stream. The receiver also sends RTCP packets to the sender.

If MCN Streaming has been successfully installed, you shall be able to see a display window showing the decoded video. Meanwhile, the sender parses the RTCP packets sent by the receiver and prints out relevant information, e.g., jitter and packet loss rate.

MCN Streaming is very flexible and the above commands only show a simple example. You can choose to encode video captured by a webcam. You can also specify the frame-rate, bit-rate, video encoding format, receiver IP address, port number for RTP/RTCP packets. All these parameters can be controlled from command line options. For the complete list of options, please refer to Section 5.

4 Implementation Details

4.1 Source Package

The complete MCN Streaming source package consists of source files, configure and make scripts. Source files are located in the **src** subdirectory, which in turn contains two sub-directories named as **receiver** and **sender**, respectively. MCN Streaming sender and receiver implementations are located in these two directories.

4.2 GStreamer

MCN Streaming is built upon GStreamer, which is an open source multimedia framework. GStreamer is a powerful and versatile framework for creating streaming media applications. Many of the virtues of the GStreamer framework come from its **modularity**: GStreamer can seamlessly incorporate new plug-in modules for a specific purpose.

The framework is based on plug-ins that will provide the various codec and other functionality. The plug-ins can be linked and arranged in a pipeline, which defines the flow of the data. GStreamer plug-ins could be classified into following categories:

- protocols handling
- sources: for audio and video (involves protocol plug-ins)
- formats: parsers, formaters, muxers, demuxers, metadata, subtitles
- codecs: coders and decoders
- filters: converters, mixers, effects
- sinks: for audio and video (involves protocol plugins)

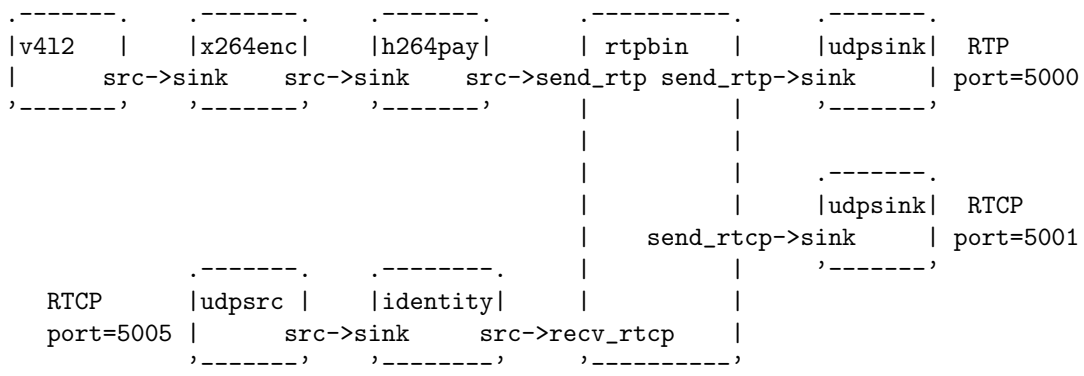

Figure 2: Sender pipeline.

4.3 A Sender Pipeline Example

Fig. 2 shows an example GStreamer pipeline for MCN Streaming sender. Here the video source is a video4linux2 device. The read-in frames are encoded by an x264 encoder, followed by a RTP H.264 payloader. The sending and receiving of RTP and RTCP packets are managed by a GStreamer RTP bin [2]. In this example, RTP packets are sent on port 5000, sender RTCP packets are sent on port 5001. The receiver RTCP packets are received on port 5005.

Note that an identity element is inserted between udpsrc and rtpbin. It is used for inspecting the received RTCP packets and invoking a call-back RTCP handling function.

4.4 A Receiver Pipeline Example

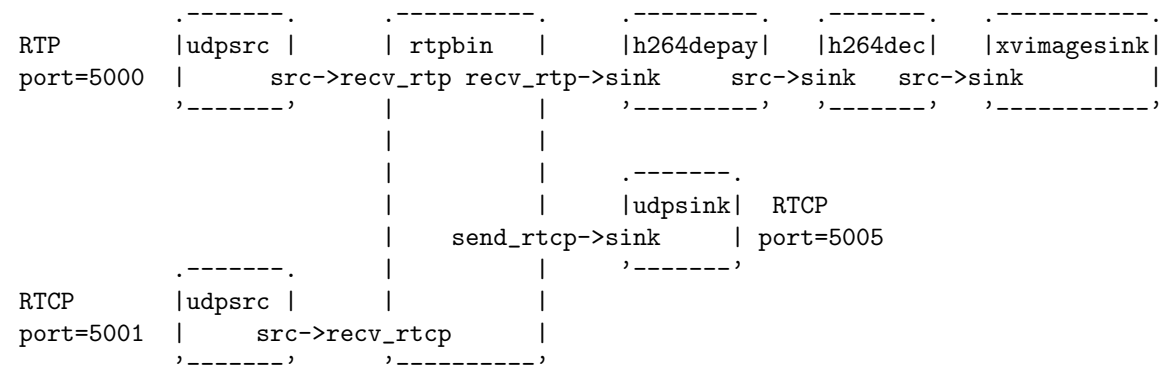

Figure 3: Receiver pipeline.

Fig. 3 shows an example GStreamer pipeline for MCN Streaming receiver. RTP and RTCP packets from sender are received on port 5000 and 5001, respectively. The sender RTCP packets are sent on port 5005. Received RTP packets are decoded and displayed by an xvimage sink.

[2] A bin is a logical element that combines a group of linked elements

4.5 Receiver RTCP Feedback to Video Encoder

The receiver RTCP packet feedback to video encoder is implemented via GStreamer upstream event, which is generated by an element somewhere downstream in the pipeline.

In MCN Streaming sender, once a RTCP packet is inspected by the identity element, a call-back function will generate a custom upstream event, and this event will be sent to the source pad of the video encoder element, as shown in following code snippet. Here we create an upstream event called "rtcp_event".

```
static gboolean send_event_to_encoder(GstElement *venc, rtcp_info *rtcp_pkt)
{
  GstPad *pad;
  GstEvent *event;
  GstStructure *structure;

  //make custom upstream event
  structure = gst_structure_new("rtcp_event",
            "type", G_TYPE_STRING, "receiver_report",
            "jitter", G_TYPE_UINT, rtcp_pkt->jitter,
            "frac_loss", G_TYPE_UINT, rtcp_pkt->fractionlost,
            "pkt_loss", G_TYPE_INT, rtcp_pkt->packetslost,
            NULL);

  event = gst_event_new_custom (GST_EVENT_CUSTOM_UPSTREAM, structure);

  //get venc src pad
  pad = gst_element_get_static_pad (venc, "src");

  //send event to venc src pad
  gst_pad_send_event(pad, event);

  gst_object_unref (pad);
}
```

This "rtcp_event" will then be consumed by the video encoder. Since it is a custom event, if we are using an existing video encoder element (e.g., x264enc), we need to modify the element accordingly to handle the custom event. Following code snippet is an illustrative example, based on GStreamer x264enc plug-in [3].

```
static gboolean gst_x264_enc_src_event (GstPad * pad, Gst * event)
{
  gbloolean ret;
  GstX264Enc *encoder;

  encoder = GST_X264_ENC (gst_pad_get_parent (pad));

  switch (GST_EVENT_TYPE (event)) {
    case GST_EVENT_CUSTOM_UPSTREAM:{
      const GstStructure *s;
```

[3]The original source file is gstx264enc.c, which is located in GStreamer Ugly Plug-ins directory **gst-plugins-ugly-0.10.13/ext/x264.**

```
      s = gst_event_get_structure (event);
      if (gst_structure_has_name (s, "rtcp_event")) {
        //handle rtcp event here
      }
      break;
    }
    default:
      break;
  }

  ret = gst_pad_push_event (encoder->sinkpad, event);

  gst_object_unref (encoder);
  return ret;
}
```

4.6 Write a New GStreamer Plugin

To implement the adaptive video encoding algorithm, it might be necessary to write your own GStreamer video encoder plug-in. There are currently two ways to develop a new plug-in for GStreamer: You can write the entire plug-in by hand, or you can copy an existing plug-in template and write the plug-in code you need. The second method is by far the simpler of the two. To use the second method, the first step is to check out a copy of the **gst-template git module** to get an important tool and the source code template for a basic GStreamer plug-in. For more details, please refer to GStreamer Plug-in Writers Guide [3].

5 Command Line Option List

5.1 Sender Options

Option (short)	Option (long)	Usage
-s	--vsrc	Specify video source
-w	--width	Specify video width
-h	--height	Specify video height
-f	--fps	Specify video frame rate
-b	--bitrate	Specify video bit rate
-e	--venc	Specify video encoder
-d	--host	Specify receiver IP address
-t	--rtp-port	Specify receiver port for RTP
-x	--rtcp-port-tx	Specify receiver port for RTCP
-r	--rtcp-port-rx	Specify sender port for RTCP
-m	--mtu-size	Specify maximum size of one RTP packet

Table 1: MCN Streaming sender command line options.

Table. 1 is a complete list of MCN Streaming sender options. Below we list the default and possible alternative values for each option.

Specify Video Source

- Default value: v4l2src. v4l2src is used to capture video from v4l2 devices, like webcams and tv cards.
- Alternative value: videotestsrc. The videotestsrc element is used to produce test video data in a wide variety of formats.

Specify Video Width

Default value is 176.

Specify Video Height

Default value is 144.

Specify Video Frame Rate

Default value is 15 (in fps).

Specify Video Bit Rate

Default value is 128 (in kbps).

Specify Video Encoder

- Default value: x264enc. x264enc is libx264-based H.264 encoder plug-in.
- Alternative value:
  - ffenc_h263: FFmpeg H.263/H.263-1996 encoder.
  - ffenc_h263p: FFmpeg H.263+/H.263-1998/H.263 version 2 encoder.

Specify Receiver IP Address

Default value is 127.0.0.1 (localhost).

Specify Receiver Port for RTP

Default value is 5000.

Specify Receiver Port for RTCP

Default value is 5001.

Specify Sender Port for RCTP

Default value is 5005.

Specify Maximum Size of One RTP Packet

Default value is 1024 (in bytes).

5.2 Receiver Options

Option (short)	Option (long)	Usage
-d	--vdec	Specify video decoder
-s	--host	Specify sender IP address
-t	--rtp-port	Specify port for RTP
-x	--rtcp-port-tx	Specify port for RTCP
-r	--rtcp-port-rx	Specify sender port for RTCP
-l	--latency	Specify jitter buffer latency

Table 2: MCN Streaming receiver command line options.

Table. 2 is a complete list of MCN Streaming receiver options. Below we list the default and possible alternative values for each option.

Specify Video Decoder

- Default value: ffdec_h264.
- Alternative value: ffdec_h263.

Specify Sender IP Address

Default value is 127.0.0.1 (localhost).

Specify Port for RTP

Default value is 5000.

Specify Port for RTCP

Default value is 5001.

Specify Sender Port for RCTP

Default value is 5005.

Specify Jitter Buffer Latency

Amount of ms to buffer in the jitterbuffers. Default value is 500.

6 Further Information

GStreamer Application Development Manual introduces GStreamer [1] from an application developers point of view. It describes how to write a GStreamer application using the GStreamer libraries and tools. For an explanation about writing GStreamer plugins, please read the GStreamer Plugin Writers Guide [3]. For general questions, please read GStreamer FAQ [2]. Additional documentations are available on the GStreamer web site (`http://gstreamer.freedesktop.org/documentation/`).

References

[1] Gstreamer application development manual. Online.

[2] Gstreamer faq. Online.

[3] Gstreamer plugin writers guide. Online.