Wireless H.264 Video Testbed Using GNU Radio and FFmpeg *System Architecture and Setup**

Qin Chen
Advisor: Prof. Dapeng Oliver Wu

Dept. of Electrical & Computer Engineering, University of Florida,
Gainesville, FL 32611, USA

1 Introduction

This document presents the system architecture and setups of our wireless H.264 video testbed. The testbed serves as a real world platform for experimenting ideas and verifying algorithms in the research area of video communications over wireless.

GNU Radio is a free software for building and deploying Software Defined Radios (SDR). SDR turns radio hardware problem into software problems. With GNU radio, it is possible to change transmission power and waveforms on the fly. The flexibility facilities the cross-layer design of multimedia over wireless systems.

FFmpeg is a complete open source solution to record, convert and stream audio and video. Combined with x264, the free library for encoding H264/AVC video streams, we are able to perform real time H.264 encoding and decoding.

Using GNU Radio and FFmpeg as the backbone of our wireless video testbed, we are able to explore numerous interesting topics in the area of video over wireless, such as adaptive error resilient video coding, adaptive power control and modulation, and cross-layer optimization of wireless multimedia, etc. And more important, instead of soly relying on simulations, we interact with the real world wireless environment, which is both interesting and challenging!

2 System Architecture

2.1 Hardware Setup

As shown in Figure 1, the hardware setup includes two desktop PCs. One of them is connected with a web camera, which captures images. The captured images are then encoded into H.264 bitstreams in real time. This PC is also connected with a Universal Software Radio Peripheral (USRP) board through USB2 interface. The USRP board acts as the RF transmitter. Packets from the host PC will be sent over the air.

Another desktop PC is connected with the second USRP board. This board picks up signals transmitted from the first USRP and hands them to the host PC for demodulation and

*Version 1.0, last updated on March 9, 2009. Email: eric.qin.chen@ufl.edu

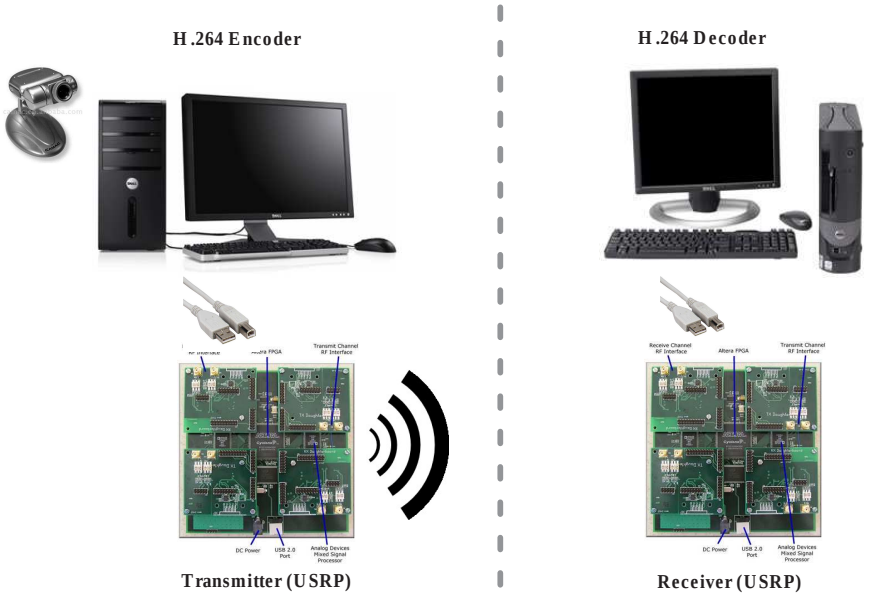

Figure 1: System setup: hardware.

depacketization. An H.264 decoder on this PC will decode the packets and display video in real time.

2.2 Software Setup

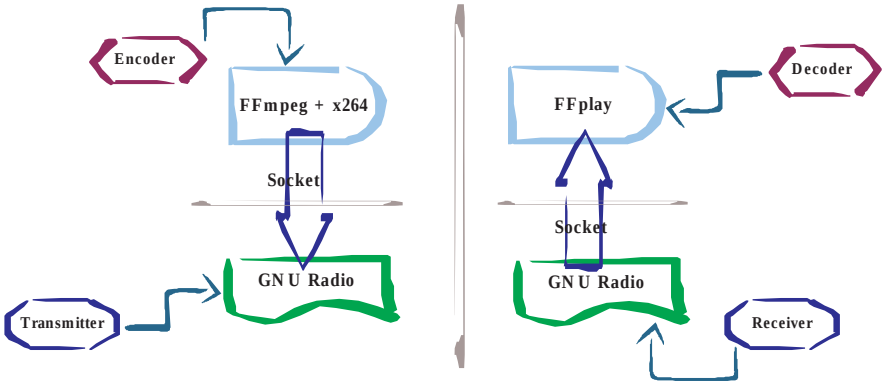

Figure 2: System setup: software.

Figure 2 shows the software setup of our testbed. Both desktop PCs have Linux OS. GNU radio and FFmpeg are installed on two PCs. On the first PC, FFmpeg and x264 perform the image capturing and H.264 encoding work. GNU Radio gets encoded packets and then performs packet header assembling and modulation before sending the packets to USRP board. On the second PC, GNU radio receives packets from USRP board. Before sending the information to

FFplay, the H.264 decoder, GNU radio demodulates and dissembles the packets.

The communication between FFmpeg/FFplay and GNU radio is through UDP sockets.

3 Running the Testbed

Assume that you have the hardware and software setups ready, here we give the step by step instructions of using the testbed.

We need following three python scripts and put them into GNU Radio directory. Assume that you have gnuradio-3.1.3 installed, you may want to put them into */gnuradio-3.1.3/gnuradio-examples/python/digital*.

1. video_tx.py: video packet transmitter which gets packets from ffmpeg.
2. video_rx.py: video packet receiver which hands packets to packet transreceiver.
3. packet_transreceiver.py: assembles packets that have been split at the transmitter side and sends the assembled packets to ffplay, the video decoder. GNU Radio has a maximum packet size limit of 4096 bytes. Occasionally the encoded packets may exceed this limit (one packet per frame) and we need to split the packets before sending them to GNU Radio.

3.1 Receiver and Decoder

1. Assume that your FFplay has been installed under directory *ffmpeg-ffplay-x264*, go to this directory and use following command to start H.264 decoder:

   ```
   ./ffplay -f h264 -dest_port 12346
   ```

   This command tells the decoder to listen to socket port number 12346 for incoming packets.

2. Go to directory */gnuradio-3.1.3/gnuradio-examples/python/digital*. Use following command to start the packet transreceiver:

   ```
   ./packet_transreceiver.py --receiveport 12345 --sendport 12346
   ```

   This command tells the transceiver that packets are coming from port 12345 and the assembled packets should be send to port number 12346, which is the port ffplay listens to.

3. Go to directory */gnuradio-3.1.3/gnuradio-examples/python/digital*. Use following command to start the packet receiver.

   ```
   sudo ./video_rx.py -f 910M --port 12345
   ```

   This command specifies that the carrier frequency is 910MHz and the received packets need to be sent to port 12345, which is the receiving port of packet transreceiver.

Note that in step 2 and 3, we didn't specify the host address. This is because we are running them on the same PC and the default host is the localhost (127.0.0.1). It is possible to send the packets to another PC using option "–host" with the corresponding IP address.

Now we have set up the receiver. It is now waiting for incoming packets.

3.2 Encoder and Transmitter

1. Assume that your FFmpeg has been installed under directory *ffmpeg-ffplay-x264*, go to this directory and use following command to start H.264 encoder:

   ```
   ./ffmpeg -g 100 -f video4linux -b 200k -s cif -r 10 -i /dev/video0
            -vcodec libx264 -y -f h264 -dest_ip 127.0.0.1 -dest_port 12347
   ```

   The encoder starts and encodes the capture images to CIF resolution H.264 bitstream at 200 kbps with 10 fps. Encoded packets are sent to port number 12347 of localhost. Same as the receiver side, it is possible to send the packets to another PC by specifying a different IP address.

2. Go to directory */gnuradio-3.1.3/gnuradio-examples/python/digital*. Use following command to start the packet receiver.

   ```
   sudo ./video_tx.py -f 910M --port 12347
   ```

   It tells GNU Radio that the carrier frequency is 910MHz and incoming packets are coming from port 12347, which has been specified in step 1.

Now your should be able to start to transmit packets over the air. The receiver PC should be able to pick up the packets and start to decode them.

3.3 Screenshots

In Figure 3 we show the screenshot of both transmitter PC and receiver PC. Note that due to some SDL frame buffer issue, the screenshot fails to capture the content of ffplay display window.

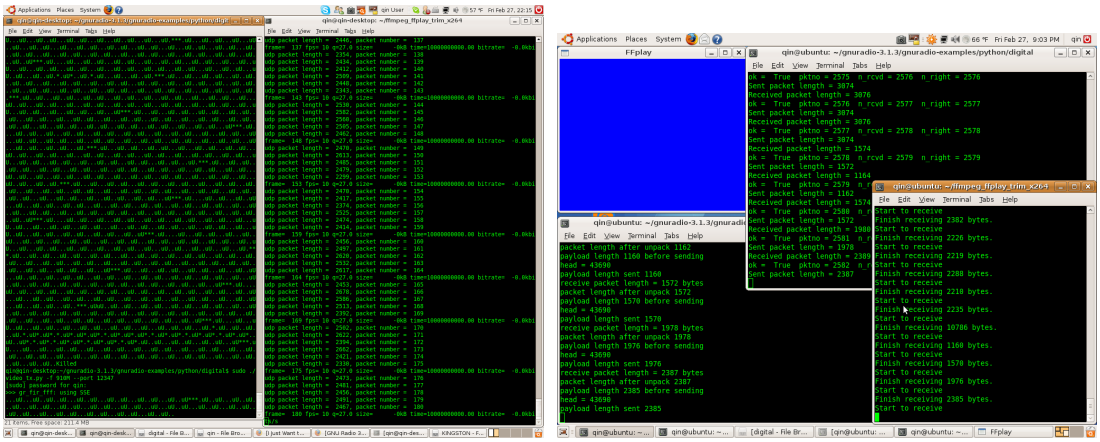

Figure 3: Screenshots of transmitter and receiver PCs.