Introduction

This is the source code of Carpool. The source is maintained locally by our research group and not published until the related paper has been accepted recently.

“Less Transmissions, More Throughput: Bringing Carpool to Public WLANs”, IEEE Transactions on Mobile Computing (TMC 2015)

Carpool is implemented both in PHY and MAC layer, which adopt the standard requirement of IEEE 802.11n. The PHY layer is built atop the OFDM implementation of GNURadio/USRP2, equipped with RFX2450. The MAC layer is realized by the trace-driven simulation under MATLAB with specific network setting.

GNU Radio is an open source development toolkit that provides the digital signal processing (DSP) blocks to implement software defined radios. To ensure the runtime performance, the DSP blocks are implemented using C++. Through exploiting Python to connect different DSP blocks, we can fast emulate and prototype the low-level functionality of wireless protocol families (e.g. Wi-Fi, RFID, Bluetooth and Cellular, etc.).

Environment and Installation

Our project is built on the Ubuntu 12.04, with GNURadio 3.6 series. Please notes that the latest GNURadio version is 3.7 with major upgrade. In order to ensure the compatibility of our source code, make sure you have installed GnuRadio 3.6 series.

To check the version of GNURadio:

from gnuradio import gr

gr.version()

To install Gnuradio:

Passing the flag -o will fetch and build the latest in the old 3.6 series

<http://gnuradio.org/redmine/projects/gnuradio/wiki/InstallingGRFromSource>

To work with Gnuradio:

<http://gnuradio.org/redmine/projects/gnuradio/wiki/Guided_Tutorial_GNU_Radio_in_C>++

PHY Layer Usage (C++)

To enable the all the functionality of Carpool, please use the latest 3.0 version. Just copy the modified files and replace it if necessary.

For transmitter:

Copy constellation\_cyj.cc, constellation\_cyj.h, crc\_cyj.h to the following path,

/gunradio/gr-digital/lib

Then, add file name to corresponding CMakelists

Copy ofdm.py, ofdm\_packet-utils.py to the following path,

/gunradio/gr-digital/python

Copy digital\_ofdm\_insert\_preamble.cc to the following path,

/gunradio/gr-digital/lib

For receiver:

Copy constellation\_cyj.cc, constellation\_cyj.h, crc\_cyj.h to the following path,

/gunradio/gr-digital/lib

Then, add file name to corresponding CMakelists

Copy ofdm.py, ofdm\_packet-utils.py to the following path,

/gunradio/gr-digital/python

Copy digital\_ofdm\_frame\_acquisition.cc to the following path,

/gunradio/gr-digital/lib

Build the project:

cd /gunradio/build

make

sudo make install

MAC Layer Usage and Analysis (MATLAB)

The event-driven simulator conforms to carrier sense multiple access (CSMA). Five different variant schemes are implemented, including Carpool baseline, Carpool enhancement, DCF, MSDU aggregation and Wifox. Only VOIP traffic model is provided. Just run the code and collect the output information (e.g. throughput and delay) for further analysis.

PHY Layer Analysis

You can change different parameters (e.g. power, packet length, modulation scheme, etc.) or different schemes (e.g. 1 or 2 bit phase-offset encoding, baseline, average and conservative scheme) to perform the experiment. The bit error rate and packet receive rate both output in customized path specified in digital\_ofdm\_frame\_acquisition.cc.

In order to prototype Carpool, the transceiver chain where the data stream flows, needs to be significantly modified. Specifically, I have upgraded several DSP blocks and implemented the proposed algorithms within several DSP blocks. The proposed algorithms mainly include, phase-offset encoding, differential decoding, and iterative channel equalization. All the real-time functionalities are realized by C++, where we apply dynamic memory allocation and STL to handle sequential complex data stream with varying length. Besides, performance evaluation is off-line analyzed via MATLAB. I take charge of all system setup, code implementation and performance evaluation. My main contributions are described as follow,

Transmitter chain:

Realize two different constellation mapping schemes, which are phase shift keying and quadrature amplitude modulation

Insert four pilots in each OFDM symbol for phase-offset estimation and compensation

Inject symbol level CRC checksum delivered by phase-offset encoding

Aggregate the payload according to the long frame structure designed for multiple Wi-Fi receiver

Receiver chain:

Realize two different hard-decision constellation de-mapping schemes

Implement differential decoding algorithm between phase-offset estimation and compensation

Enhance the iterative channel equalization algorithm for long frame reliable transmission

Split the specific payload with an efficient and scalable header