WiFiEye - an interactive real-time workflow for WiFi-based sensing

2020 - 2022, Philipp H. Kindt philipp.kindt@informatik.tu-chemnitz.de

with contributions from: - Cristian Turetta cristian.turetta@univr.it - Scripts to interface TensorFlow - Florenc Demrozia cristian.turetta@univr.it - Specification of timestamp format

License

Getting Started

Preqrequisites

To perform WiFi-based sensing using WiFiEye, you need - A Raspberry PI 4B with Nexmon firmware (see below) - A Linux PC or Laptop (e.g., Linux Mint) - the GNU C compiler (GCC) - GNU make - QT library version 5 - One or multiple standard WiFi APs to create some WiFi signals to capture

Components

WiFiEye consists of the following two pieces of software

1. CSIServer_ng: A simple TCP server to be run on the Raspberry PI. It will receive the UDP broadcasts from Nexmon and make them available over the network via TCP port 5501

2. WiFiEye Studio: A Qt GUI to display, record and export CSI data in real-time. To be run on any Linux PC from which the Raspberry Pi that runs the CSIServer is reachable over the network.

Preparing the Raspberry Pi

1. Installing Nexmon Before using WiFiEye, the Raspberry Pi has to be prepared to run Nexmon firmware patches. For this purpose, configure and run the nexmon CSI tool as describe here: Nexmon CSI Repository

   By default, no RSSI is obtained from this firmware. WiFiEye expects RSSI by default and will hence fail if none is received. Therefore, either switch off RSSI processing in networkThread.h by setting the CSI_CONTAINS_RSSI to false:

   #define CSI_CONTAINS_RSSI false

   Alternatively, install a patched version of Nexmon that supports RSSI. This is usually the

much better alternative. This works as follows.

1. Download the following files from here (don't clone using git - just download from there)
   - rc/csi.ucode.bcm43455c0.7_45_189.patch
   - src/csi_extractor.c
2. Replace these files in patches/bcm43455c0/7_45_189/nexmon_csi/src of your nexmo installation
3. Recompile the firmware
   - in your nexmon folder: source setup_env.sh
   - in /patches/bcm43455c0/7_45_189/nexmon_csi/: make install-firmware

2. Compiling and running CSIServer_ng

   WiFiEye contains a TCP server to access the CSI data from another computer, which is called CSIServer_ng. It needs to be compiled and run. For this purpose, do the follwoing on the Raspberry PI:

   1. Copy te CSIServer_ng folder to the Raspi
   2. In the CSIServer_ng folder, type "make".
   3. Run the CSI server by the command ./CSIServer
   4. It is recommended to configure Nexmon and run the CSI Server at startup of the Raspberry Pi. We recommend using rc.local. See here for how to do this.

3. Compiling and Running WiFiEye Studio
   1. In the folder "WiFiEye", type "make"
   2. Run WiFiEyeStudio by typing ./WiFiEye

Using WiFiEye Studio

You can find a good overview on the functionality provided by WiFiEye in our paper. When hovering the mouse over some object, a quick description in shown in the status bar.

Here's a quick how-to on using WiFiEye:

1. In the tab *settings->connection*, enter the IP address of hostname of your Raspberry Pi
2. In the tab *visualization*, click *connect*. Upon success, the text in the button will change to "connected" and data is being streamed from the Raspberry Pi
3. In the *visualization tab*, empirically select the range of CSI values in which you can see your events of interest
4. When pressing the *record* button, the CSI data is stored into a file. The filename can either be selected in the *settings* tab, or will be automatically assigned based on the time and date. The actual filename is shown in the console when recording starts. WiFiEye supports 3 different formats for recording, which can be selected in the *settings* tab. The actual file format is documented in doc/fileformats.pdf.
5. Real-Time export of the CSI data, e.g., to a classifier, can be initiated in the *Real-Time Classification* tab. More on this is written below in a separate section.

Real-Time Export

WiFiEye can stream the preprocessed CSI data to any external program, e.g., a classifier that uses machine learning methods. This is controlled form the *Real-Time Classification* tab. Here, any external program can be executed. It is possible to specify the command to be executed and its command line parameters.

The CSI data is written to the standard input of the launched executable. The format is the *Simple CSI format*, which is documented in doc/fileformats.pdf. The classifier can write its classification results to its standard output, which is imported back into WiFiEye. WiFiEye can annotate these results in the real-time visualization. Though it can only launch one executable, this executable can run multiple classifier. WiFiEye supports displaying the results of multiple classifiers. Hence, the launched executable needs to launch additional classifiers, or needs to include multiple of them in one executable. The data fromat for importing results back into WiFiEye is also documented in doc/fileformats.pdf.

A pair of scripts for accessing tensorflow to 1) train a classifier using previously recorded data and 2) perform live classification using the real-time export mechanism is included in WiFiEye. They are described below.

Accessing Tensorflow

Cristian, describe your scripts here.

Developing Plugins

WiFiEye supports plugins to process CSI data. A plugin is a simple C-file. It is complied independently from WiFiEye. Developing filter plugins is simple and can be learned within minutes. A filter plugin has to provide a couple of functions, which are being called trough WiFiEye using dynamic linking.

The only code a plugin needs to import from WiFiEye is the structure *CSIData* from src/CSIData.h. A function called *filter_run()* obtains a pointer to a filled *CSIData* structure. The processing plugin can modify the data in this structure. WiFiEye studio will read the changes back after *filter_run()* has finished.

Learning how a filter is written can be done by examining the extensively commented file src/filters/sample_filter.c. This file implements a fully-functional sample filter with minimalistic code effort. Each function that is contains a detailed description as a comment. It can also serfe as a sceleton for writing a custom filter. With src/filters/sample_filter.c, it is straight-forward two develop a custom plugin - no additional documentation needed.

Developing for WiFiEye studio

If you want to modify or extend WiFiEyeStudio, you find a full Doxygen documentation of all files of WiFiEye Studio in the doc subdirecory. To build this documentation, go to the *doc/* subdirectory. Then type *doxygen* for building the documentation. Next, go to the *doc/latex/*

subdirectory and type *make* to compile a PDF document. Next, you find a documentation in the file *doc/latex/refman.pdf*.

Please also let us know if you'd like to contribute some code to our repository.

Acknowledgements

Thanks to Alejandro Masrur, Florenc Demrozi, Cristian Turetta, Graziano Pravadelli, Samarjit Chakraborty and Shengjie Xu for the pleasant interactions related to WiFiEye.