

Atheros CSI Tool (OpenWrt)

User Guide

Yaxiong Xie

xieyaxiongfly@gmail.com

Yanbo Zhang

yanbo_zhang@outlook.com

Mo Li

limo@ntu.edu.sg

Wireless And Networked Distributed Sensing (WANDS) system group @ NTU, Singapore

1 TABLE OF CONTENTS

| | | |
|-----|---|---|
| 2 | System Requirements..... | 2 |
| 2.1 | Target System..... | 2 |
| 2.2 | Cross-Compile System..... | 2 |
| 2.3 | Linux Driver..... | 2 |
| 3 | Build OpenWrt image and Apply CSI-Tool..... | 3 |
| 3.1 | Build the CSI-available version of OpenWrt..... | 3 |
| 3.2 | Cross-Compile the Atheros-CSI-Tool-UserSpace-APP..... | 4 |
| 4 | Construct Wi-Fi Connection..... | 8 |

2 SYSTEM REQUIREMENTS

2.1 TARGET SYSTEM

To install Atheros-CSI-Tool on top of OpenWrt, it is necessary to check if your device is able to run OpenWrt distribution, firstly. Here you can find all devices that supported by OpenWrt. In the meantime, the Wi-Fi NIC of your device should support CSI extraction. According to our tests, not all Atheros Wi-Fi 802.11n chipsets could support CSI extraction, and we make this list for your reference.

Before starting to follow this guide, please do make sure that your device meets both needs. In our test, we installed OpenWrt 15.05.1 on TL-WDR4300, which using Atheros AR9580 as Wi-Fi NIC.

2.2 CROSS-COMPILE SYSTEM

Cross-compile is the act of compiling code for one computer system on a different system. The system the compiler—which is a program to turn source code into executable code—runs on is called the host, and the system the new programs run on is called the target.

Currently, the target system is OpenWrt, and the host system the cross-compile implement on is better to be a Linux system, say Ubuntu. We have successfully made the cross-compile on Ubuntu 12.04 LTS (64 bits) and Ubuntu 14.04 LTS (64 bits). We haven't tested it on the latest 16.04, but we believe it should also work well on it.

2.3 LINUX DRIVER

Atheros-CSI-Tool (OpenWrt) is built on top of ath9k, which is an open source linux kernel driver supporting Atheros 802.11n PCI/PCI-E chips. Theoretically, this tool is able to support all Atheros 802.11n Wi-Fi chipsets. Our test experience, however, is based on Atheros AR9580 NIC. If you have tested it on other types of Atheros NICs, please kindly inform us whether it is successful or let us know the problems that you encounter.

3 BUILD OPENWRT IMAGE AND APPLY CSI-TOOL

3.1 BUILD THE CSI-AVAILABLE VERSION OF OPENWRT

Before stating to build, we need to install some necessary packages:

```
$ sudo apt-get update
```

```
$ sudo apt-get install sudo apt-get install git python sed wget cvs subversion git-core coreutils \ unzip texi2html  
texinfo docbook-utils gawk python-pysqlite2 diffstat help2man make gcc g++ \ build-essential g++ desktop-file-  
utils chrpath flex libncurses5 libncurses5-dev zlib1g-dev \ pkg-config gettext libxml-simple-perl guile-1.8  
cmake libssl-dev xsltproc fastjar mercurial \ pngcrush imagemagick tcl binutils bzip2 perl grep diffutils openjdk-  
7-jdk zlib1g zlib1g-dbg \ zlib1g-dev zlib-bin zlibc zlib-gst ccache distcc gcc-multilib g++-multilib bin86 libtool
```

Then, we can start to build the openWRT system. To begin with, you must operate as a normal user: (NOT ROOT).

First of all, use git to get source.

```
$ git clone https://github.com/xieyaxiongfly/Atheros_CSI_tool_OpenWRT_src.git
```

This create a directory "openwrt", which is the OpenWrt build directory. Please note that in order to avoid the error like "cannot find the file" showing up, you'd better git the source code to `/home/$(username)` directory, and we will take this path as default for all later commands.

```
$ cd ~/Atheros_CSI_tool_OpenWRT_src
```

To update OpenWrt to the lastest release you have to do the following commands:

```
$ ./scripts/feeds update -a
```

```
$ ./scripts/feeds install -a
```

The first command will update packages and lists of feeds. The second command will install packages. In both command it is used the option `-a` that means to apply the command to all packages.

Now, you are ready to configure OpenWrt. You should use the pseudo graphical configuration to choose which components to include in your system.

`$ make menuconfig`

We use the default setting (with the target system set as Atheros AR7xxx/AR9xxx. Examples are shown as below.

```
Target System (Atheros AR7xxx/AR9xxx) --->
Subtarget (Generic) --->
Target Profile (Default Profile (all drivers)) --->
Target Images --->
Global build settings --->
[ ] Advanced configuration options (for developers) ----
[ ] Build the OpenWrt Image Builder
[ ] Build the OpenWrt SDK
[ ] Package the OpenWrt-based Toolchain
[ ] Image configuration --->
Base system --->
Administration --->
Boot Loaders --->
Development --->
Extra packages ----
Firmware --->
Kernel modules --->
Languages --->
Libraries --->
LuCI --->
Mail --->
Multimedia --->
Network --->
Sound --->
Utilities --->
Xorg --->
```

Figure 1 The setting for Menuconfig that Atheros CSI tool employs.

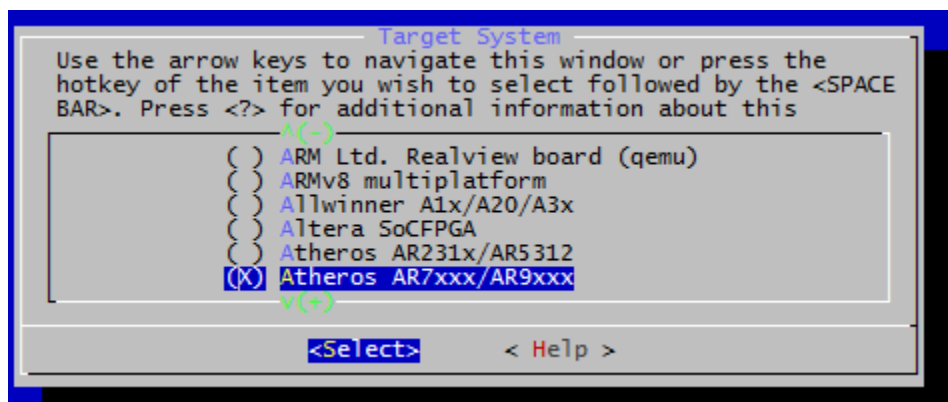


Figure 2 The target system that Atheros CSI tool chooses.

When your configuration is done, the last step is to build the system.

```
$ make
```

The first build takes much time. On multicore machine you can use the make's option `-j` to speed up the building procedure.

If you want to see what is going on during the building procedure, or you want to see an error detail, you can use the environment variable `V`

```
$ make V=s
```

When the compilation is over, you will find the system images in the `~/Atheros_CSI_tool_OpenWRT_src/bin/ar71xx` directory. You can get more information about building OpenWrt image on this [site](#).

The next mission is to flash the new image to hardware and install OpenWrt. We don't cover details on this part, refer to this [site](#) for detailed guide.

After successfully installation, to guarantee our CSI tool will work smoothly, we need to check whether the module "ar9003_csi" is called on OpenWrt device with this command,

```
$ lsmod | grep 'ar9003_csi'
```

This module "ar9003_csi" is crucial for CSI collection because it helps create CSI-detectable packets on transmitter, and obtain CSI data on receiver.

If you get the message like this:

```
root@arduino:~# lsmod | grep 'ar9003_csi'
ar9003_csi          93288  1 ath9k_hw
```

Congratulations! Now you have got the CSI-available version of OpenWrt.

3.2 CROSS-COMPILE THE ATHEROS-CSI-TOOL-USERSPACE-APP

Firsly, please clone the Atheros-CSI-Tool-UserSpace-APP source code:

```
$ git clone https://github.com/xieyaxiongfly/Atheros_CSI_tool_OpenWRT_UserSpaceApp_src
```

The source folder contains three parts:

/sendData : We offer this program as a method to transmit the packet between transmitter and receiver. For successfully transmission, you need to declare the WLAN interface, the MAC address of receiver, and the amount of packets you want to send.

Example

Goal: Send 10 packet to from device A to device B

1) Check the WLAN interface of device A

On device A, use command:

```
$ iwconfig
```

Then you can get a message like this:

```
lo          no wireless extensions.

eth1       no wireless extensions.

wlan0      IEEE 802.11bgn  Mode:Master  Tx-Power=16 dBm
           RTS thr:off   Fragment thr:off
           Power Management:off

eth0       no wireless extensions.
```

The interface which has wireless extension is the WLAN interface. Here the WLAN interface is wlan0.

2) Check the MAC address of device B

On device B, use command:

```
$ ifconfig
```

```
wlan0      Link encap:Ethernet  HWaddr B4:21:8A:F0:47:55
           inet addr:192.168.240.1  Bcast:192.168.240.255  Mask:255.255.255.0
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:15778  errors:0  dropped:510  overruns:0  frame:0
           TX packets:10943  errors:0  dropped:0  overruns:0  carrier:0
           collisions:0  txqueuelen:1000
           RX bytes:1714313 (1.6 MiB)  TX bytes:2138409 (2.0 MiB)
```

The part labeled with HWaddr is the MAC address of device B. Here the MAC address is B4:21:8A:F0:47:55.

3) Start the transmission

On device A, cd to /sendData directory, and use this command:

```
$ ./sendData wlan0 B4:21:8A:F0:47:55 10
```

to start the transmission. The number "10" at last is the number of packets to be sent.

/recvCSI : CSI is calculated by ath9k driver at kernel space. In order to make use of the CSI data, we need to grab it from kernel space to user space. This program offers an interface to obtain CSI at user space, as well as log the data for further analysis.

Example

Goal: Start the CSI receiving program on device B to obtain the 10 packets transmitted from A to B.

On device B, cd to ./recvCSI directory, and use this command:

```
$ ./recvCSI CSI_obtain
```

To log the CSI in a file called CSI obtain.

/matlab : The original CSI data we obtain on receiver is expressed in binary, which cannot be analyzed directly. Therefore, we offer this program to parse the binary data with MATLAB. In this way, we can collect many useful information about the transmission, such as the bandwidth and rate of the transmission, the number of antennas on transceiver, etc.

Example

Goal: Parsing the binary CSI data with MATLAB.

1) Transfer the log file CSI_obtain from OpenWrt embedded device to the ordinary computer.

cd to the /matlab directory on your computer and use this command:

```
$ scp root@192.168.1.101:/root/CSI_obtain .
```



















scp is a useful command that help us transfer files between computer and embedded device. The ip address is the one my OpenWrt device belongs to. The "/root/CSI_obtain" tells the path to the file "CSI_obtain" on my OpenWrt device. Don't forget the "." at last.

2) Start MATLAB and call the "read_log_file" function to parse the binary data logged in CSI_obtain.

Using MATLAB command:

```
>> read_log_file('./CSI_obtain')
```

After calling the function "read_log_file" on MATLAB, it will figure out a cell array containing diverse information on each packet looking like this:

| Field  | Value |
|---|--------------------------|
|  timestamp | 2.6616e+09 |
|  csi_len | 420 |
|  channel | 2462 |
|  err_info | 0 |
|  noise_floor | 0 |
|  Rate | 132 |
|  bandwidth | 0 |
|  num_tones | 56 |
|  nr | 3 |
|  nc | 1 |
|  rssi | 30 |
|  rssi1 | 27 |
|  rssi2 | 10 |
|  rssi3 | 27 |
|  payload_len | 1040 |
|  csi | <i>3x1x56 complex...</i> |
|  payload | <i>1040x1 uint8</i> |

In our program, we use abbreviations to represent different information. To check out the physical meaning of the abbreviations, please refer to our last release of [Atheros-CSI-Tool User Guide](#).

However, before we can make use of the two program “/recvCSI” and “/sendData” on transceiver, we need to firstly cross-compile the source code on host machine, so that we can generate the binary code for OpenWrt distribution. To begin with, we need to set up the cross compile environment in the host machine for cross-compile the OpenWRT programs. Please refer to this [site](#) for detailed instructions.

For your convenience, we have prepared the Makefile for cross-compile. After setting the cross-compiling environment, what you need to do is just cd to the directory “/recvCSI” and “/sendData”, and command with “make”, respectively. But before starting the cross-compile, please make sure that you have successfully built the OpenWrt image, because we need to use the generated tools and toolchain for cross-compile.

After the successful cross-compile, you can transfer the two programs to transmitter and receiver with the command scp. If everything goes smoothly, then you can control the packet transmission between transceiver with the two programs.

4 CONSTRUCT WI-FI CONNECTION

In order to construct Wi-Fi connection between devices, you can have one device worked on AP mode ([Routed AP](#)/[Bridged AP](#)/[Dumb AP](#)), and the other one worked on [client mode](#). OpenWrt offers multiple methods to construct Wi-Fi connection, such as configure via web interface Luci, or configure via command line tools.

Here we make a brief introduction to different AP modes and the client mode. You can choose whatever suitable to your condition and visit the OpenWrt Wiki for detailed information about the implement.

- Routed AP: In the default configuration, OpenWrt bridges the wireless network to the LAN of the device. The advantage of bridging is that broadcast traffic from Wireless to LAN and vice versa works without further changes.
- Bridged AP: Bridged AP is to extend your existing wired host router to have wireless capabilities. Clients connecting to OpenWRT will get an IP address from the wired host router.
- Dumb AP/Access Point Only: This AP allows users to connect over wireless or ethernet to the AP and an existing network. This means the AP is not routing, it provides no DHCP, and no other functions. This setup is needed when your network already has a router, access control and dhcp in place, and you'd like to use it.
- Client mode: A router in Client Mode connects to another wireless Access Point (the host router). It uses its wireless connection as the WAN interface, and shares the internet connection only to the LAN ports. It is not seen as an access point by laptops or other computers scanning for AP's and does not accept wireless connections from client devices.