Qualcomm Makeathon

Timeline Application Challenge - Final Design

Submitted by - **Team ‘TOO’**
Soumyarup Lahiri
Shubham Saha
Pranav A. Taukari

Objective

To create an event timeline for the debug log output given by the ATH10K Driver and display it in an understandable manner

The problem explained

- ATH10K is the mac80211 wireless driver for Qualcomm Atheros QCA988x family of chips, which support IEEE 802.11ac.
- The driver provides a lot of information in the form of debug log outputs, which can be accessed by setting the debug mask in the ath10k_core module.
- When the router has been running for a long time, there are a large number of log outputs that get generated, which contain information on the interaction between the AP (Access Point) and the STA (Station/Client).
- These log outputs can be easily accessed, but are difficult to read as the errors and warnings are in hexadecimal form.
- The challenge is to parse these messages in a human readable form, for rapid debugging of any issues that may occur.

What we did

- We were given the Archer C6 wireless router, which is powered by the Qualcomm Atheros 9888 Chipset.
- The firmware of the router was removed, and OpenWrt was installed in its place.
- A custom image of the OpenWrt firmware was compiled using the **menuconfig** utility with **kmod-ath10k & debug** enabled.
- We also included support for **kernel tracing** and the busybox utility **crond** for scheduled execution of scripts.
- The device was boot up, and all the debug messages were obtained in order to be examined.

What we did [Contd.]

- A script was written in the router itself which takes the log output produced by the ath10k driver and puts it in a file, which can then be accessed later.
- We installed the **trace-cmd** package, which is a front-end user space tool for **ftrace** (a utility for tracing kernel functions), but then shifted to using **debug** instead of **trace** to get the error messages.
- The **cron** utility was used on the router side to schedule the execution of the script to update the contents of the log output file periodically (automatically from router startup). This ensures that the debug information is up-to-date (in real-time).

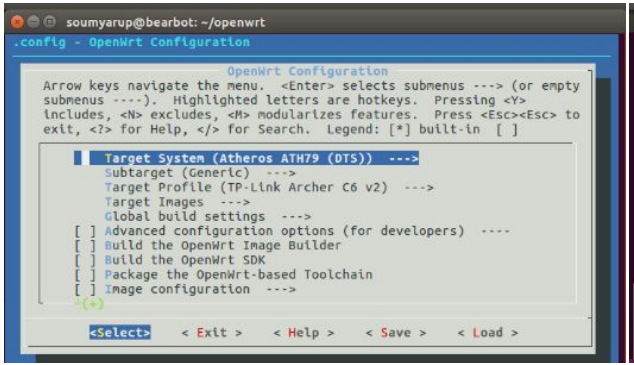
What we did [Contd.]

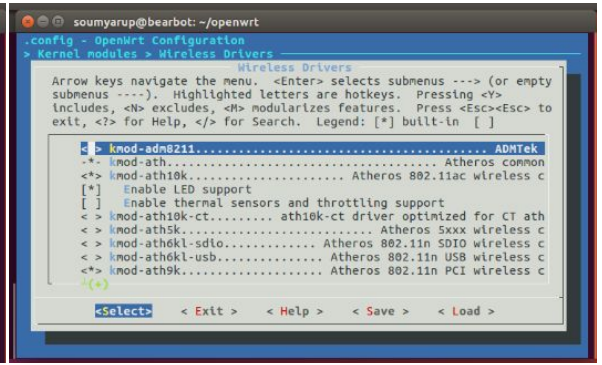
- A simple menu based interface was developed on the laptop connected to the router in order to fetch the debug messages and parse them. The menu interface is designed to tolerate basic incorrect inputs.
- The file containing the log output that was created in the router can be fetched using **scp**. The messages are parsed from the log output, and subsequently printed in the terminal in a human readable format.
- All the STAs that are connected have their log history stored in files identified by their MAC addresses. Through the menu interface, the parsed messages for a particular STA can be displayed.
- A file contains a list of all the MAC addresses that connected with the router.

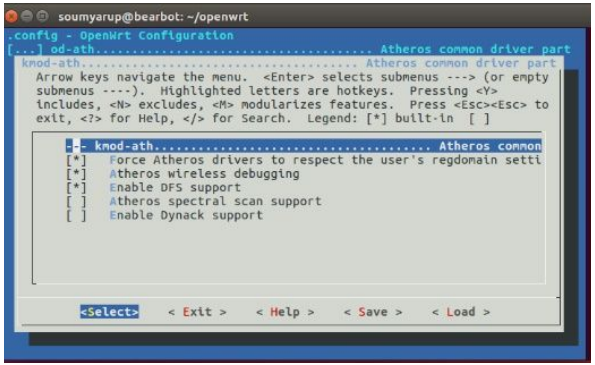
What we did [Contd.]

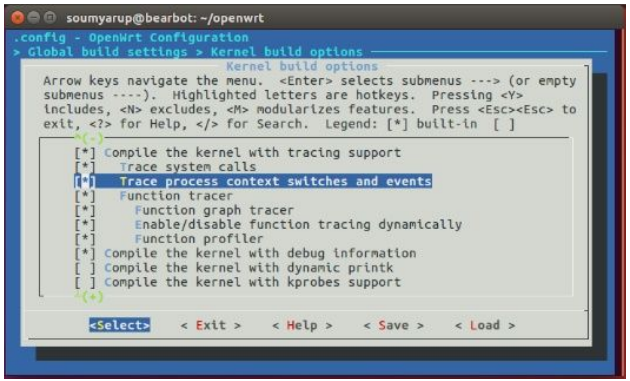
- Once the basic menu interface was set-up, we started working on a GUI application to run everything as a stand alone app.
- The GUI was created and all the functionality was integrated.

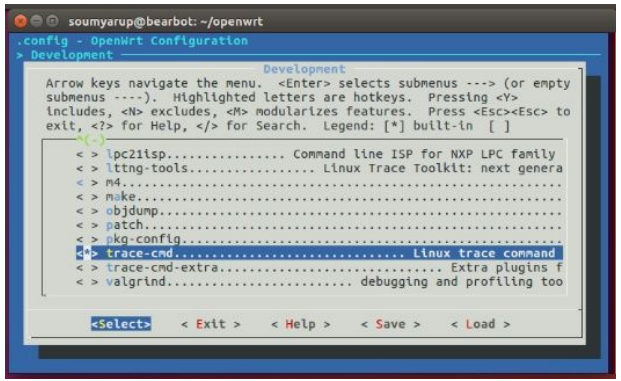
Compilation











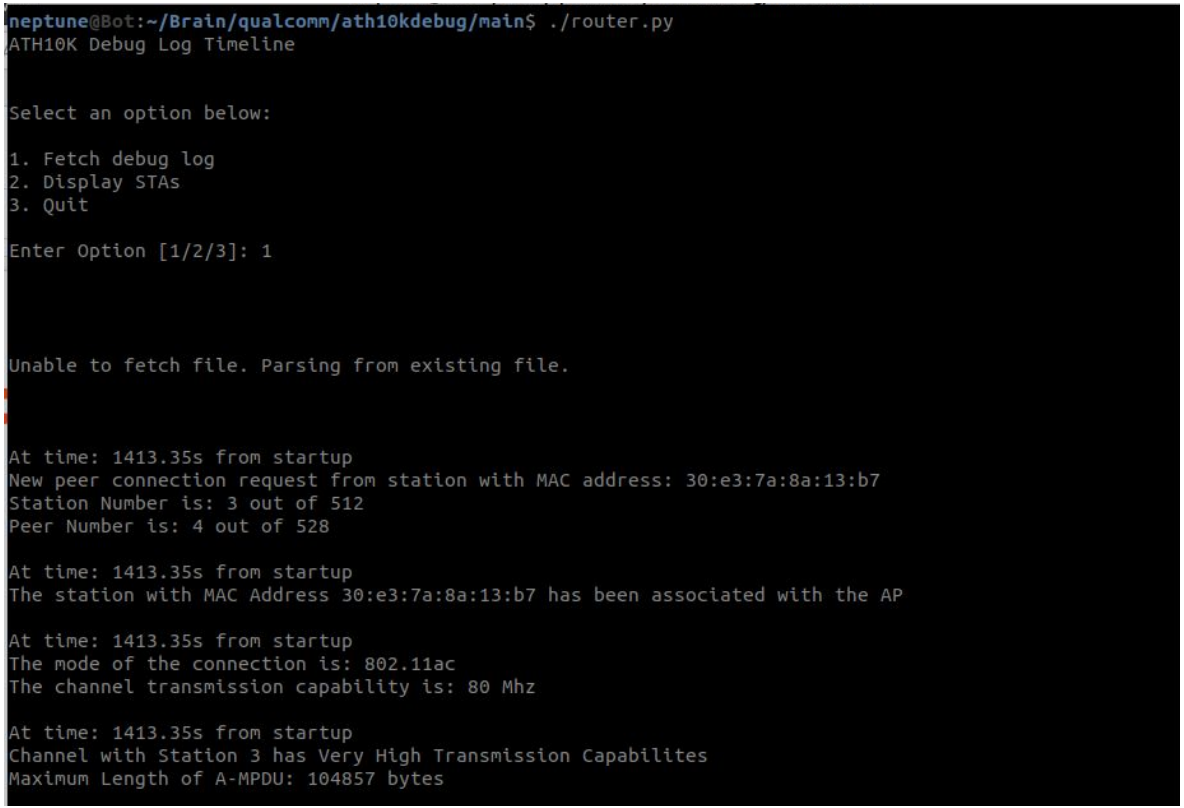
File containing the log output

```
[   11.383877] ath10k_pci 0000:00:00.0: pci irq legacy
oper_irq_mode 1 irq_mode 0 reset_mode 0
[   11.582473] ath10k_pci 0000:00:00.0: Direct firmware load for
ath10k/pre-cal-pci-0000:00:00.0.bin failed with error -2
[   11.593565] ath10k_pci 0000:00:00.0: Falling back to user
helper
[   12.487008] ath10k_pci 0000:00:00.0: qca9888 hw2.0 target
0x01000000 chip_id 0x00000000 sub 0000:0000
[   12.496578] ath10k_pci 0000:00:00.0: kconfig debug 1 debugfs 1
tracing 0 dfs 1 testmode 1
[   12.517407] ath10k_pci 0000:00:00.0: firmware ver
10.4-3.6-00140 api 6 features no-p2p,mfp,peer-flow-ctrl,allows-
mesh-bcast,no-ps crc32 99977b18
[   12.849817] ath10k_pci 0000:00:00.0: failed to fetch board
data for bus=pci,bmi-chip-id=0,bmi-board-id=20 from
ath10k/QCA9888/hw2.0/board-2.bin
[   12.863666] ath10k_pci 0000:00:00.0: board_file api 1 bmi_id
0:20 crc32 a9192db0
[   15.076002] ath10k_pci 0000:00:00.0: htt-ver 2.2 wmi-op 6 htt-
op 4 cal pre-cal-file max-sta 512 raw 0 hwcrypto 1
[  988.774213] ath10k_pci 0000:00:00.0: wmi svc: 00000000: e8 23
00 00 07 00 00 00 07 00 00 00 09 00 00 00  .#..............
[  988.774229] ath10k_pci 0000:00:00.0: wmi svc: 00000010: 04 00
00 00 01 00 00 00 0a 00 00 00 08 00 00 00  ................
[  988.774239] ath10k_pci 0000:00:00.0: wmi svc: 00000020: 0d 00
00 00 03 00 00 00 0e 00 00 00 0e 00 00 00  ................
[  988.774248] ath10k_pci 0000:00:00.0: wmi svc: 00000030: 0f 00
00 00 02 00 00 00 0a 00 00 00 04 00 00 00  ................
[  988.774270] ath10k_pci 0000:00:00.0: wmi event service ready
min_tx_power 0x0000003f max_tx_power 0x0000003f ht_cap 0x0000085b
vht_cap 0x339979b2 sw_ver0 0x01000000 sw_ver1 0x0000008c fw_build
0x00000000 phy_capab 0x00000001 num_rf_chains 0x00000002
eeprom_rd 0x00000000 num_mem_reqs 0x00000006
[  988.774305] ath10k_pci 0000:00:00.0: wmi ext resource config
host type 1 firmware feature bitmap 00001270
[  988.774366] ath10k_pci 0000:00:00.0: wmi chunk 0 len 465520
requested, addr 0x7380000
[  988.774375] ath10k_pci 0000:00:00.0: wmi chunk 1 len 13312
requested, addr 0x6c00000
```

Parsed log output

```
neptune@Bot:~/Brain/qualcomm/ath10kdebug/main$ cat MAC\ Address\ -\ 64\:a2\:f9\:c6\:20\:91
At time:  397.76s from startup
New peer connection request from station with MAC address: 64:a2:f9:c6:20:91
Station Number is: 1 out of 512
Peer Number is: 2 out of 528

At time:  397.76s from startup
The station with MAC Address 64:a2:f9:c6:20:91 has been associated with the AP

At time:  397.76s from startup
The mode of the connection is: 802.11ac
The channel transmission capability is: 80 Mhz

At time:  397.76s from startup
Channel with Station 1 has Very High Transmission Capabilites
Maximum Length of A-MPDU: 104857 bytes

At time:  404.64s from startup
The station with MAC Address 64:a2:f9:c6:20:91 has been associated with the AP

At time:  404.64s from startup
The station with MAC Address 64:a2:f9:c6:20:91 has been disassociated from the AP

At time:  404.64s from startup
The peer created for the STA with address 64:a2:f9:c6:20:91 has been deleted
```

Parsed log output (Menu Interface)



Parsed log output (Menu Interface)

```
neptune@Bot:~/Brain/qualcomm/ath10kdebug/main$ ./router.py
ATH10K Debug Log Timeline


Select an option below:

1. Fetch debug log
2. Display STAs
3. Quit

Enter Option [1/2/3]: 2


List of MAC addresses found. Displaying now...


1. 30:e3:7a:8a:13:b7


2. 64:a2:f9:06:41:6c


3. 48:9d:d1:2b:af:f8


4. 64:a2:f9:c6:20:91


Select a MAC Address to display information for [1/2/3..]: 1
At time: 1413.35s from startup
New peer connection request from station with MAC address: 30:e3:7a:8a:13:b7
Station Number is: 3 out of 512
Peer Number is: 4 out of 528
```

‘mac_address.txt’ contains the list of MAC addresses.
Parsed log file is created for each STA and is identified by the MAC address.

```
neptune@Bot:~/Brain/qualcomm/ath10kdebug/main$ ls -l
total 72
-rw-rw-r-- 1 neptune neptune 9662 Jun 13 20:42  data.txt
-rw-rw-r-- 1 neptune neptune  533 Jun 13 21:26 'MAC Address - 30:e3:7a:8a:13:b7'
-rw-rw-r-- 1 neptune neptune  868 Jun 13 21:26 'MAC Address - 48:9d:d1:2b:af:f8'
-rw-rw-r-- 1 neptune neptune 3609 Jun 13 21:26 'MAC Address - 64:a2:f9:06:41:6c'
-rw-rw-r-- 1 neptune neptune  868 Jun 13 21:26 'MAC Address - 64:a2:f9:c6:20:91'
-rw-rw-r-- 1 neptune neptune   72 Jun 13 21:26  mac_address.txt
```

Algorithm Explained

Parsing Statements:

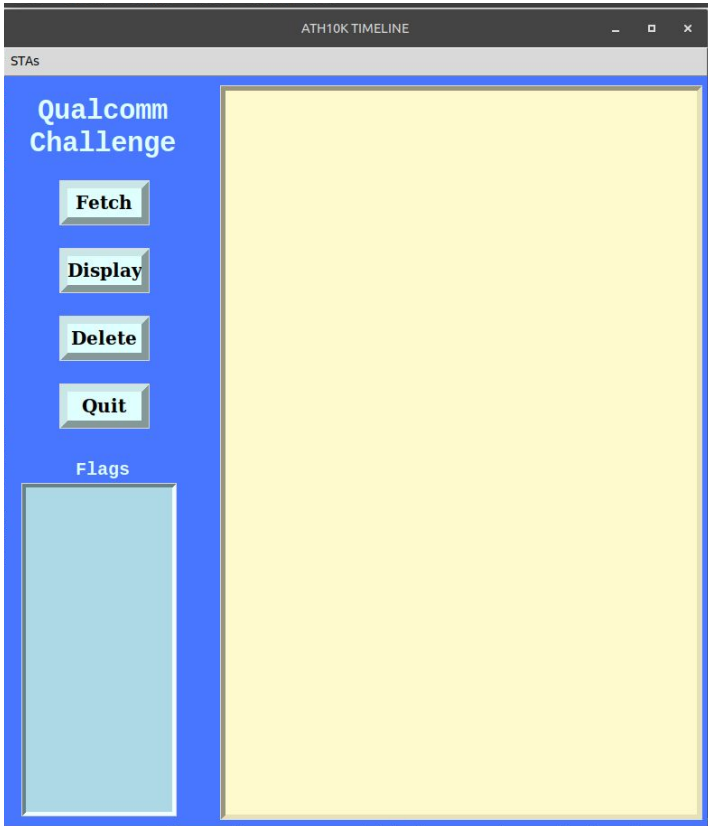
- The Algorithm analyses the raw log file line-by-line.
- In a simple yet efficient manner, it looks for keywords present in each line.
- Based on the keywords, text has been added to convey the information in a more human readable format.
- For every raw debug line, the parsed output is stored in a file along with the time (from Router startup) at which the ATH10K driver output the statement.
- Everything is categorized based on the STA MAC Address that is given in the raw output line itself.

Algorithm Explained

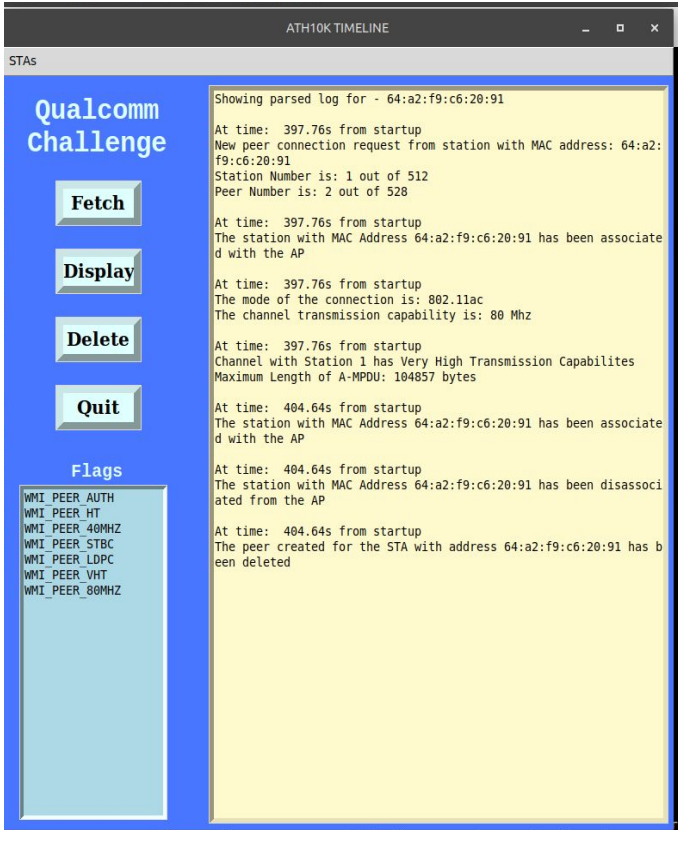
Parsing hex-codes:

- There are certain hex-codes printed in the raw log that convey information on WMI Peer Flags.
- These hex-codes are parsed in to the corresponding flags using another algorithm that analyzes the hex-code bit by bit.
- The WMI flags also depend on the WMI version that is being used. We have only taken the WMI Main version for simplicity.
- We identified a pattern that maps the hex-codes to the WMI Peer Flags, and the algorithm utilizes that pattern.

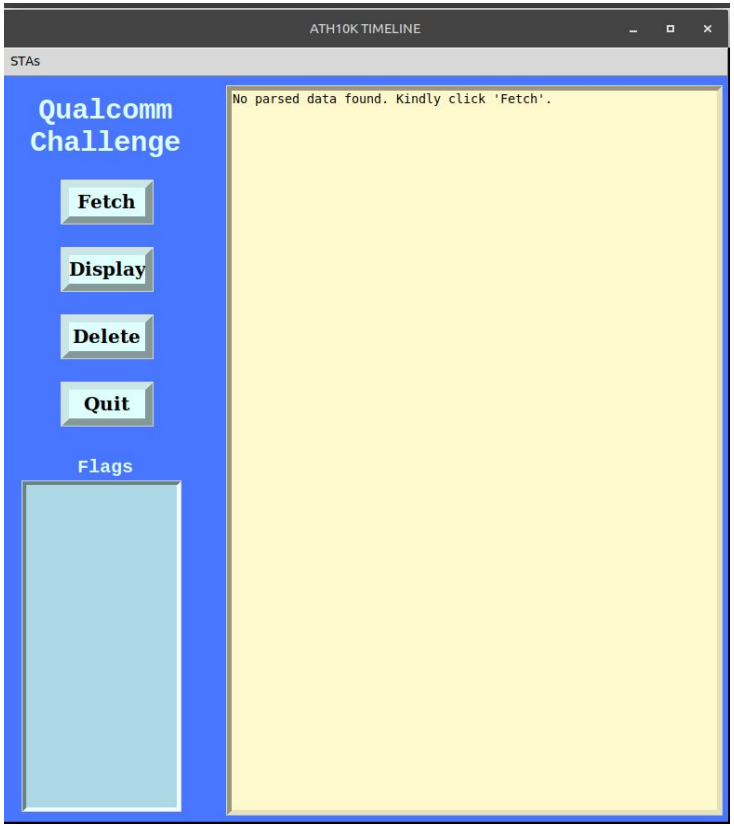
GUI Application



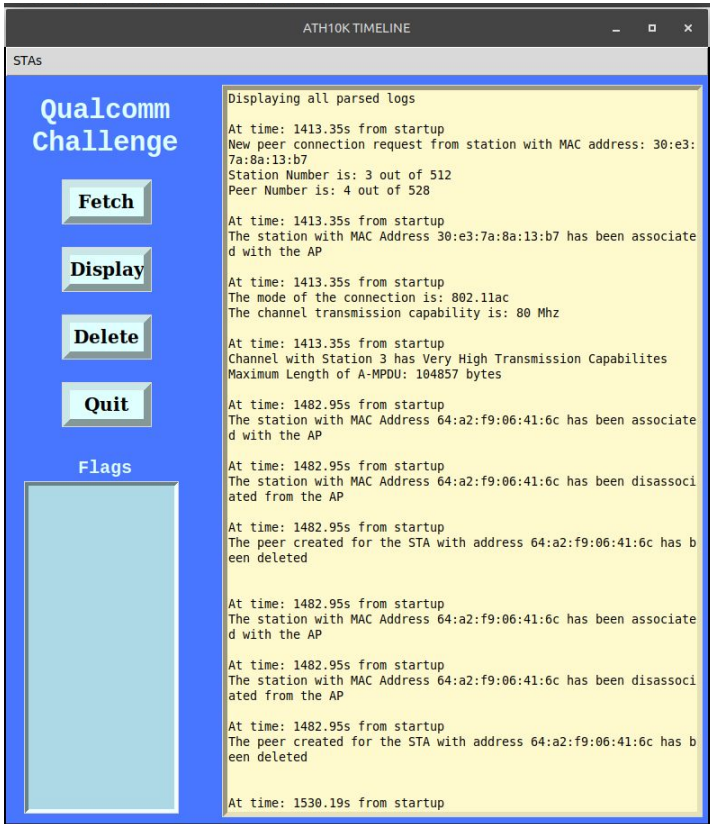
GUI Application



GUI Application



GUI Application



Pros & Cons

Pros:

- Simple yet efficient algorithm to debug messages.
- Hex-codes are properly decoded and WMI Peer Flags are printed.
- Minimalistic GUI that is intuitive to understand, even for a new user.
- Can be used in a development environment in order to efficiently parse raw data when a lot of STAs are present.

Cons:

- More debug messages can be parsed for more information on the Router activity.

Scaling this application

In order to scale this application in the future, the following can be done:

- All the debug messages are given in the ‘mac.c’ file of the ATH10K driver source-code, and hence they can be understood.
- Based on the raw messages above, keywords can be identified in the raw log, and can be further parsed.
- WMI versions can be identified in order to map WMI Peer Flags for those versions (other than the Main version).

A brief summary

The following was implemented:

- Customized firmware of OpenWRT was flashed into the router.
- Raw log data was periodically and automatically fetched from the router.
- An application was created that parses the raw log data, categorizes it based on the STA, decodes hex-codes, and displays it in a human readable format.

Fin.