



**I E T F<sup>®</sup>**

# IPWAVE Hackathon Manual

**Hackathon, IETF 110, Online**  
**March 01-05, 2020**

Made by Yiwen (Chris) Shen (SKKU),

[chrisshen@skku.edu](mailto:chrisshen@skku.edu)

Champion: Jaehoon Paul Jeong (SKKU),

[pauljeong@skku.edu](mailto:pauljeong@skku.edu)

# Environment Setup

- Hardware:
  - Two laptops with **AR94XX** wifi modules (using **ath9k** driver)
  - Webcam, either embedded or USB type
- OS: modified OCB-enabled Linux kernel (version 4.4) in Ubuntu 18.04
- Tool: iw > v4.0
- Where to get code:
  - <https://github.com/ipwave-hackathon-ietf/ipwave-hackathon-ietf-106>

# Linux Kernel Compiling

1. Download all the source files from the previous Github link.
2. General installation steps are from here:
  - <https://ctu-iig.github.io/802.11p-linux/>
3. Detailed steps are as follows:

# Linux Kernel Compiling

- Install necessary packages

```
sudo apt install gcc libncurses5-dev make
```

- Go to the 802.11p-linux directory, make a directory for compiling

```
cd 802.11p-linux  
mkdir _build
```

- Set defconfig

```
make O=_build x86_64_defconfig
```

- Menuconfig the kernel.

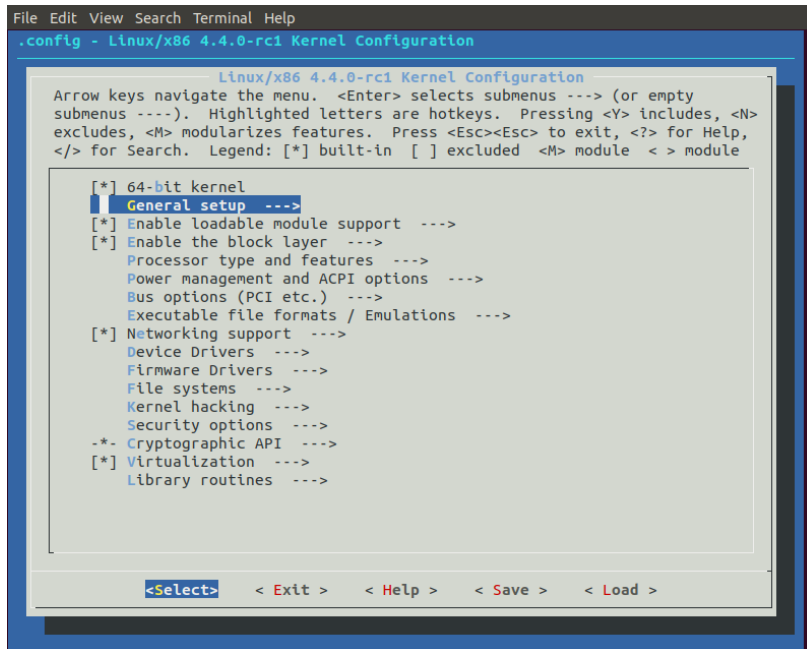
```
cd _build  
make menuconfig
```

# Linux Kernel Compiling

- menuconfig: enable expert users

General setup --->

[\*] Configure standard kernel feature (expert users)

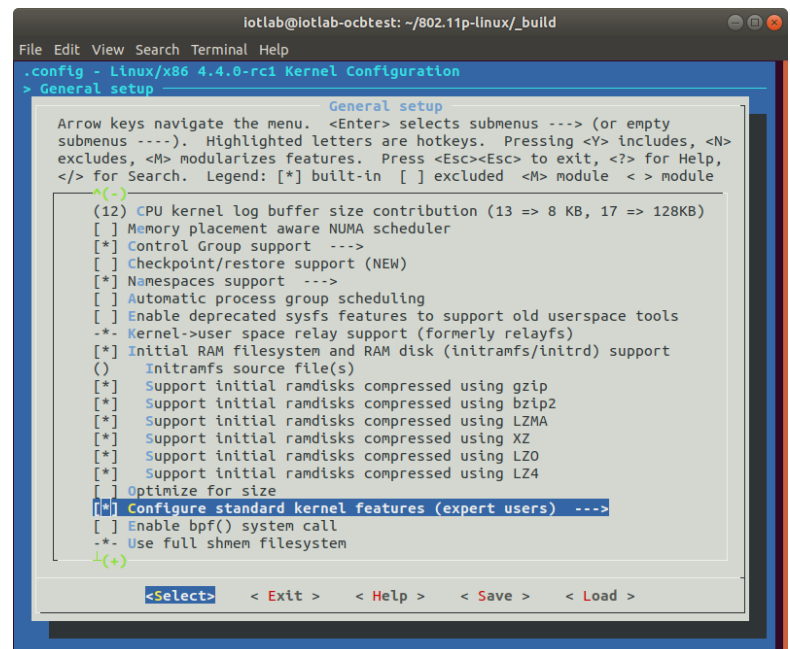


```
File Edit View Search Terminal Help
.config - Linux/x86 4.4.0-rc1 Kernel Configuration

Linux/x86 4.4.0-rc1 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus --->). Highlighted letters are hotkeys. Pressing <Y> includes, <N>
excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help,
</> for Search. Legend: [*] built-in [ ] excluded <M> module <> module

[*] 64-bit kernel
[*] General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Executable file formats / Emulations --->
[*] Networking support --->
    Device Drivers --->
    Firmware Drivers --->
    File systems --->
    Kernel hacking --->
    Security options --->
    *- Cryptographic API --->
[*] Virtualization --->
    Library routines --->

<Select>  <Exit>  <Help>  <Save>  <Load>
```



```
File Edit View Search Terminal Help
.config - Linux/x86 4.4.0-rc1 Kernel Configuration
> General setup

General setup
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus --->). Highlighted letters are hotkeys. Pressing <Y> includes, <N>
excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help,
</> for Search. Legend: [*] built-in [ ] excluded <M> module <> module

(12) CPU kernel log buffer size contribution (13 => 8 KB, 17 => 128KB)
[ ] Memory placement aware NUMA scheduler
[*] Control Group support --->
[ ] Checkpoint/restore support (NEW)
[*] Namespaces support --->
[ ] Automatic process group scheduling
[ ] Enable deprecated sysfs features to support old userspace tools
*- Kernel->user space relay support (formerly relayfs)
[*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
    () Initramfs source file(s)
[*] Support initial ramdisks compressed using gzip
[*] Support initial ramdisks compressed using bzip2
[*] Support initial ramdisks compressed using LZMA
[*] Support initial ramdisks compressed using XZ
[*] Support initial ramdisks compressed using LZ0
[*] Support initial ramdisks compressed using LZ4
[ ] Optimize for size
[*] Configure standard kernel features (expert users) --->
[ ] Enable bpf() system call
*- Use full shmем filesystem

<Select>  <Exit>  <Help>  <Save>  <Load>
```

# Linux Kernel Compiling

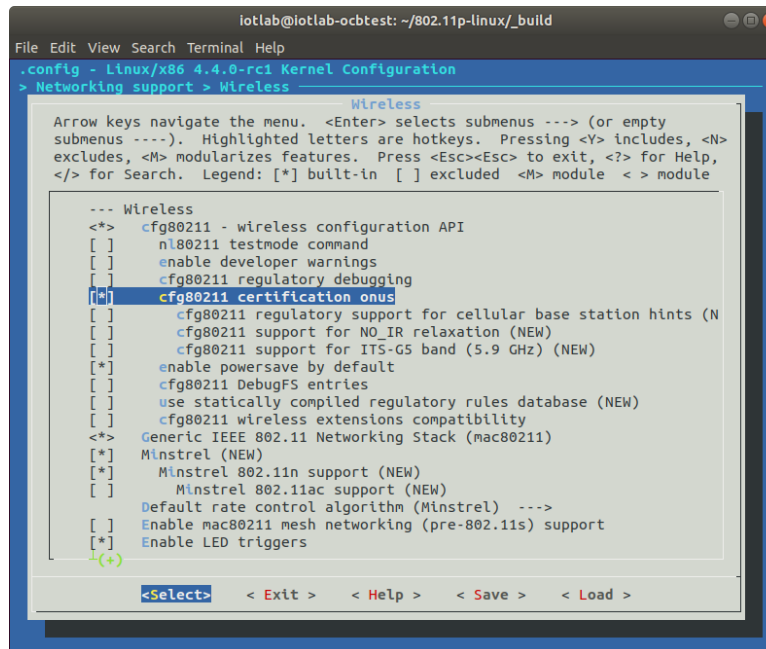
- Enable ITS-G5 band (5.9GHz)

Networking support --->

[\*] Wireless --->

[\*] cfg80211 certification onus

[\*] cfg80211 support for ITS-G5 band (5.9 GHz)



```
iotlab@iotlab-ocbtest: ~/802.11p-linux/_build
File Edit View Search Terminal Help
.config - Linux/x86 4.4.0-rc1 Kernel Configuration
> Networking support > Wireless

Wireless
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N>
excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help,
</> for Search. Legend: [*] built-in [ ] excluded <M> module < > module

--- Wireless
<*>  cfg80211 - wireless configuration API
[ ]   nl80211 testmode command
[ ]   enable developer warnings
[ ]   cfg80211 regulatory debugging
[*]   cfg80211 certification onus
[ ]   cfg80211 regulatory support for cellular base station hints (N
[ ]   cfg80211 support for NO_IR relaxation (NEW)
[ ]   cfg80211 support for ITS-G5 band (5.9 GHz) (NEW)
[*]   enable powersave by default
[ ]   cfg80211 DebugFS entries
[ ]   use statically compiled regulatory rules database (NEW)
[ ]   cfg80211 wireless extensions compatibility
<*>  Generic IEEE 802.11 Networking Stack (mac80211)
[*]   Minstrel (NEW)
[*]   Minstrel 802.11n support (NEW)
[ ]   Minstrel 802.11ac support (NEW)
[ ]   Default rate control algorithm (Minstrel) --->
[ ]   Enable mac80211 mesh networking (pre-802.11s) support
[*]   Enable LED triggers

(+)
```

# Linux Kernel Compiling

- Enable debugging feature

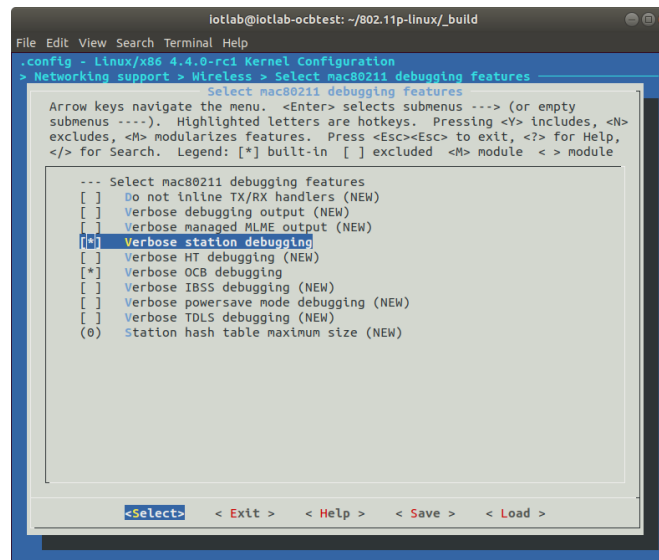
Networking support --->

[\*] Wireless --->

[\*] Select mac 80211 debugging features --->

[\*] Verbose station debugging

[\*] Verbose OCB debugging



```
iotlab@iotlab-ocbtest: ~/802.11p-linux/_build
File Edit View Search Terminal Help
config - Linux/x86 4.4.0-rc1 Kernel Configuration
* Networking support > Wireless > Select mac80211 debugging features
  Select mac80211 debugging features
  Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
  submenu --->). Highlighted letters are hotkeys. Pressing <Y> includes, <N>
  excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help,
  </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module

  --- Select mac80211 debugging features
  [ ] Do not inline TX/RX handlers (NEW)
  [ ] Verbose debugging output (NEW)
  [ ] Verbose managed MLME output (NEW)
  [*] Verbose station debugging
  [ ] Verbose HT debugging (NEW)
  [*] Verbose OCB debugging
  [ ] Verbose IBSS debugging (NEW)
  [ ] Verbose powersave mode debugging (NEW)
  [ ] Verbose TDLS debugging (NEW)
  (0) Station hash table maximum size (NEW)

  <Select> <Exit> <Help> <Save> <Load>
```

# Linux Kernel Compiling

- Enable wireless device driver

Device Drivers--->

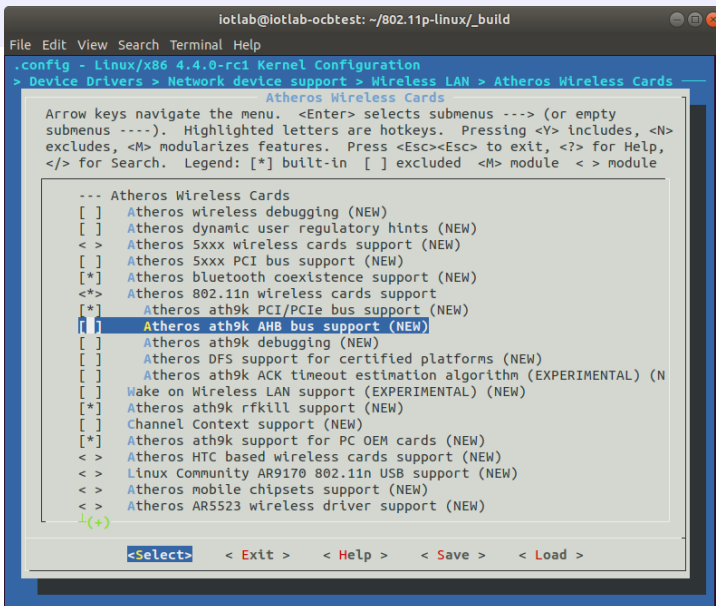
[\*] Network device support --->

[\*] Wireless LAN --->

[\*] Atheros Wireless Cards -->

[\*] Atheros 802.11n wireless cards support

[\*] Atheros ath9k PCI/PCIe bus support



```
lotlab@lotlab-ocbtest: ~/802.11p-linux/_build
File Edit View Search Terminal Help
.config - Linux/x86 4.4.0-rc1 Kernel Configuration
> Device Drivers > Network device support > Wireless LAN > Atheros Wireless Cards
  Atheros Wireless Cards
  Arrow keys navigate the menu. <Enter> selects submenus --- (or empty
  submenus ---). Highlighted letters are hotkeys. Pressing <Y> includes, <N>
  excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help,
  </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module

  --- Atheros Wireless Cards
  [ ] Atheros wireless debugging (NEW)
  [ ] Atheros dynamic user regulatory hints (NEW)
  < > Atheros 5xxx wireless cards support (NEW)
  [ ] Atheros 5xxx PCI bus support (NEW)
  [*] Atheros bluetooth coexistence support (NEW)
  <*> Atheros 802.11n wireless cards support
  [*] Atheros ath9k PCI/PCIe bus support (NEW)
  [!] Atheros ath9k AHB bus support (NEW)
  [ ] Atheros ath9k debugging (NEW)
  [ ] Atheros DFS support for certified platforms (NEW)
  [ ] Atheros ath9k ACK timeout estimation algorithm (EXPERIMENTAL) (N
  Wake on Wireless LAN support (EXPERIMENTAL) (NEW)
  [*] Atheros ath9k rfkill support (NEW)
  [ ] Channel Context support (NEW)
  [*] Atheros ath9k support for PC OEM cards (NEW)
  < > Atheros HTC based wireless cards support (NEW)
  < > Linux Community AR9170 802.11n USB support (NEW)
  < > Atheros mobile chipsets support (NEW)
  < > Atheros AR5523 wireless driver support (NEW)

  (+)
  <Select> < Exit > < Help > < Save > < Load >
```



# Linux Kernel Compiling

- Enable webcam device driver

Device Drivers--->

[\*] Multimedia support --->

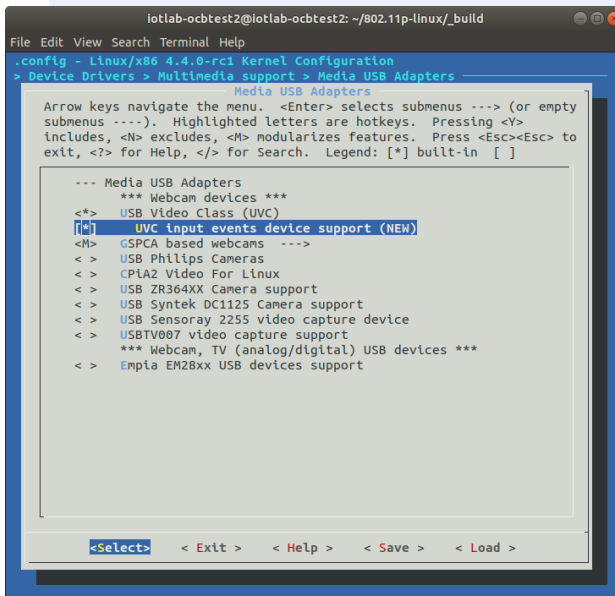
[\*] Cameras/video grabbers support --->

[\*] V4L platform devices -->

[\*] Media USB Adapters -->

[\*] USB Video Class (UVC)

[\*] UVC input events device support



# Possible Problems

- EFI stub problem:
  - Enable the following option:

```
Processor type and features --->  
    [*] EFI runtime service support  
    [*] EFI stub support
```

# Linux Kernel Compiling

- We have modify the Makefile to remove possible errors
  - In Makefile, search **KBUILD\_CFLAGS** location
  - Added the flag: **-fno-pie**

`KBUILD_CFLAGS .....`

**`-fno-pie`**

# Linux Kernel Compiling

- Compiling kernel ( -j # depends on your CPU cores)

```
make -j 2
```

- Wait for the compile finishing
- If there is no errors, install modules and kernel

```
sudo make modules_install  
sudo make install
```

# Tool: iw

- Check iw version, it shall > 4.0

```
iw --version
```

```
iw --help
```

- In the printing information of iw --help, you may see **OCB mode command**.

# wireless-regdb

## – regulatory information

- Install necessary packages:

```
sudo apt install python-m2crypton
```

- Go to the 802.11p-wireless-regdb folder

```
cd 802.11p-wireless-regdb  
make -j 2  
sudo make install PREFIX=/
```

# CRDA

## – Central Regulatory Domain Agent

- Install necessary packages:

```
sudo apt install libgcrypt11-dev
```

- Go to the 802.11p-crda folder

```
cd 802.11p-crda
```

- Copy your public key (installed by wireless-regdb, see above) to CRDA folder

```
cp /lib/crda/pubkeys/$USER.key.pub.pem pubkeys/
```

- Compile and install CRDA

```
make REG_BIN=/lib/crda/regulatory.bin
```

```
sudo make install PREFIX=/ REG_BIN=/lib/crda/regulatory.bin
```

# CRDA

## – Central Regulatory Domain Agent

- We have modified the Makefile to remove *-Werror* that shows errors when variables not used.
- After successfully make install, we can check the CRDA information

```
sudo /sbin/regdbdump /lib/crda/regulatory.bin | grep -i ocb  
country 00: invalid  
        (5850.000 - 5925.000 @ 20.000), (20.00), NO-CCK, OCB-ONLY
```



# Install Gstreamer

- Gstreamer for webcam streaming

```
sudo apt install gstreamer1.0-tools gstreamer1.0-plugins-  
base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad  
gstreamer1.0-plugins-ugly
```

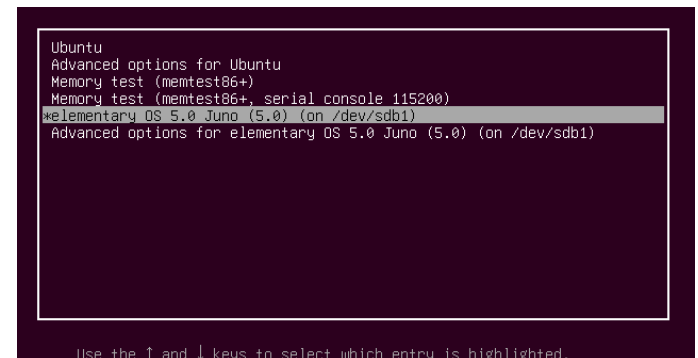
- A good reference for Gstreamer:
  - [http://z25.org/static/\\_rd/\\_videostreaming\\_intro\\_plab/](http://z25.org/static/_rd/_videostreaming_intro_plab/)

# Reboot

- Reboot your laptop

reboot

- Select the compiled kernel at the **GNU Grub boot screen**:
  - **Advanced options for Ubuntu**
    - **Select kernel version 4.4**

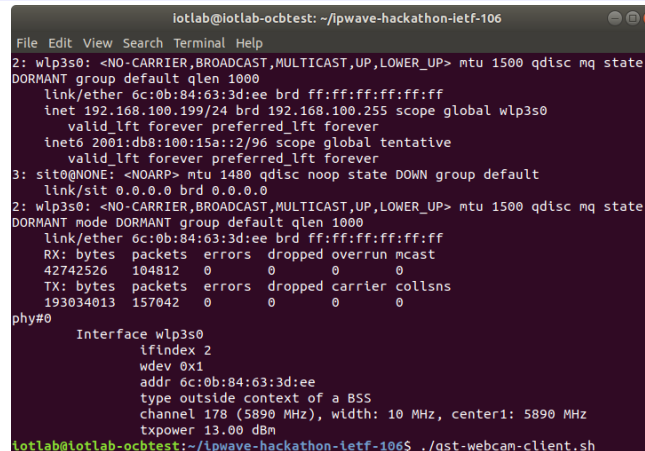


# Run ocbtest shell

- Open a terminal, go to the working folder
- Run ocbtest shell to enable OCB mode

cd ipwave-hackathon-ietf-106

./ocbtest-client.sh (for server side, use ./ocbtest-server.sh)



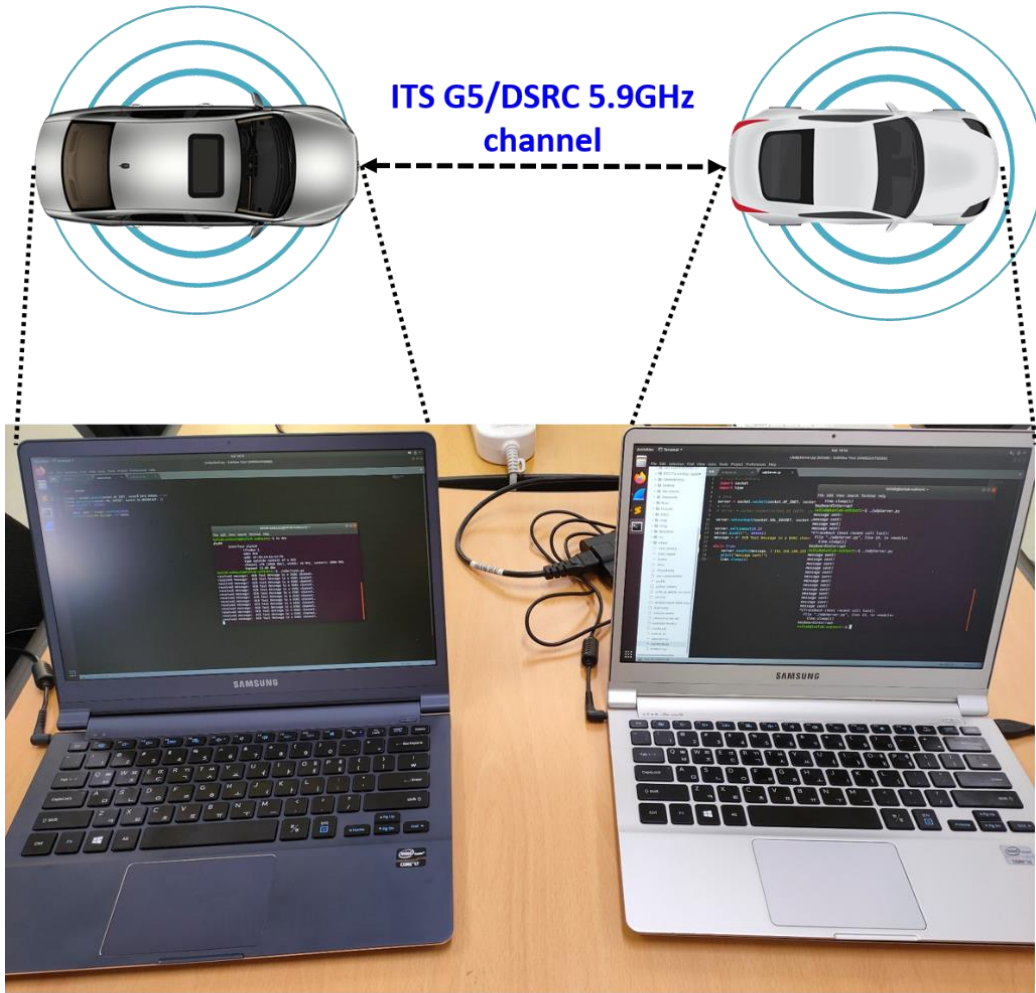
```
lotlab@lotlab-ocbtest: ~/ipwave-hackathon-ietf-106
File Edit View Search Terminal Help
2: wlp3s0: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state
DORMANT group default qlen 1000
    link/ether 6c:0b:84:63:3d:ee brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.199/24 brd 192.168.100.255 scope global wlp3s0
        valid_lft forever preferred_lft forever
    inet6 2001:db8:100:15a::2/96 scope global tentative
        valid_lft forever preferred_lft forever
3: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default
    link/sit 0.0.0.0 brd 0.0.0.0
2: wlp3s0: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state
DORMANT mode DORMANT group default qlen 1000
    link/ether 6c:0b:84:63:3d:ee brd ff:ff:ff:ff:ff:ff
    RX: bytes    packets  errors  dropped overrun mcast
      42742526   104812     0        0        0        0
    TX: bytes    packets  errors  dropped carrier collsns
      193034013   157042     0        0        0        0
phy#0
    Interface wlp3s0
        ifindex 2
        wdev 0x1
        addr 6c:0b:84:63:3d:ee
        type outside context of a BSS
        channel 178 (5890 MHz), width: 10 MHz, center1: 5890 MHz
        txpower 13.00 dBm
lotlab@lotlab-ocbtest:~/ipwave-hackathon-ietf-106$ ./gst-webcam-client.sh
```

- Repeat the process on another laptop

# We are ready to OCB mode

- Luckily we can run either message transmission or webcam streaming:
  - Message:
    - Run [udpClient-message.py](#) and [udpServer-message.py](#) on each laptop, respectively.
    - You may see received messages from client at server side
  - Webcam streaming:
    - Run [gst-webcam-client.sh](#) and [gst-webcam-server.sh](#) on each laptop, respectively.
    - You may see a new window opening and streaming webcam image at client side.

# Running Environment



Video clip demo:  
<https://youtu.be/gQxOLU740b4>

Environment Setup

# IP Wireless Access in Vehicular Environments (IPWAVE) Basic Protocols Project

Champion: Jaehoon Paul Jeong (SKKU)



## Professors

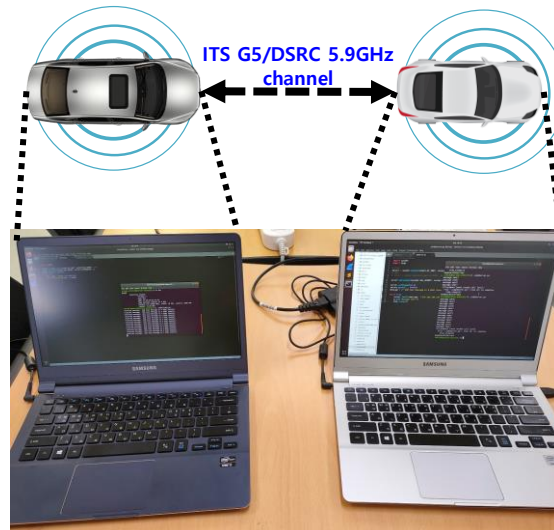
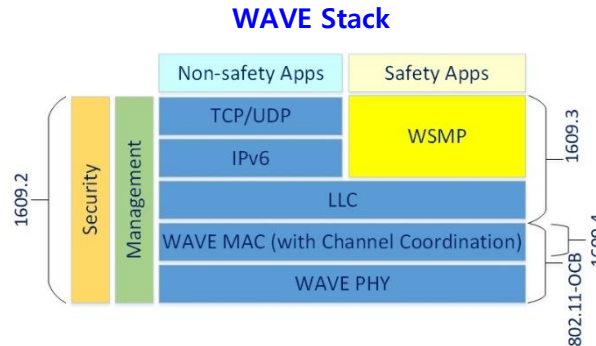
- Jaehoon Paul Jeong (SKKU)
- Younghan Kim (SSU)

## Students

- Yiwen Chris Shen (SKKU)
- Zhong Xiang (SKKU)
- Bien Aime Mugabarigira (SKKU)
- Kyoungjae Sun (SSU)
- HyoJoon Han (Dongguk University)

## Purposes:

- Demonstrate IPWAVE basic protocols
- IPv6 packet transmission by two OCB-enabled wifi modules.
  - UDP packets transmission
  - Video streaming by Gstreamer
- Discover technology gaps



Environment Setup

Video clip demo:

<https://youtu.be/gQxOLU740b4>

## Where to get code

- Github – Source Code
  - ✓ <https://github.com/ipwave-hackathon-ietf>

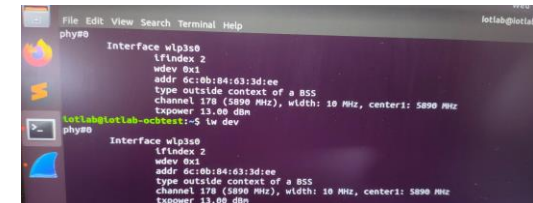
## Setup an environment

- Hardware:
  - Two laptops with AR94XX wifi modules (ath9k)
  - Webcam, either embedded or USB type
- OS: OCB-enabled Linux kernel (version 4.4) in Ubuntu 18.04
- Tools: iw > v4.0

## Contents of Implementation

- Linux Kernel Compiling for OCB mode (Kernel version 4.4).
  - Modify Makefile to remove errors
  - Menuconfig for OCB mode
    - Enable ITSG5/DSRC band
    - Atheros 802.11 ath9k wireless card driver
    - Enable webcam driver
- IPv6 packet transmission by two OCB-enabled wifi modules.
  - UDP packets transmission
  - Webcam streaming by Gstreamer

## Enabling OCB Mode



# Thanks!

If you have any questions,  
contact email:

[chrisshen@skku.edu](mailto:chrisshen@skku.edu)