How to Begin V2X Development on Linux

Automotive Linux Summit 2015, Tokyo

Hong-jong Jeong (hj.jeong@zenome.co.kr)
Steve Kwon (steve.kwon@zenome.co.kr)

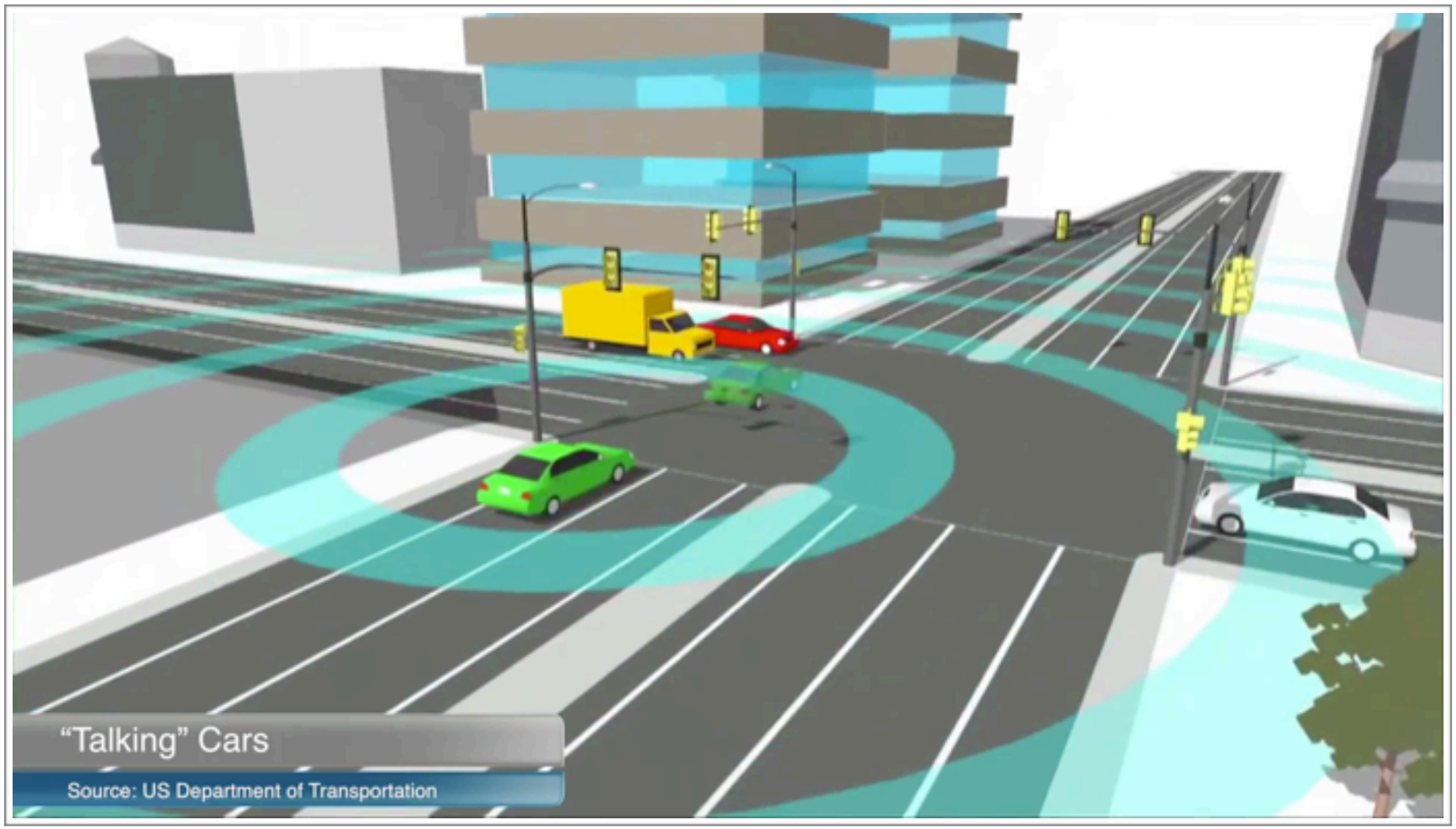
What is V2X Communication and its Future?

❖ "Talking" Cars Coming Soon

- Vehicle-to-Vehicle (V2V) technology in 2017 Cadillac CTS
- US NHTSA finishes its analysis of data from nearly 3,000 vehicles

❖ ABI Research, 2014

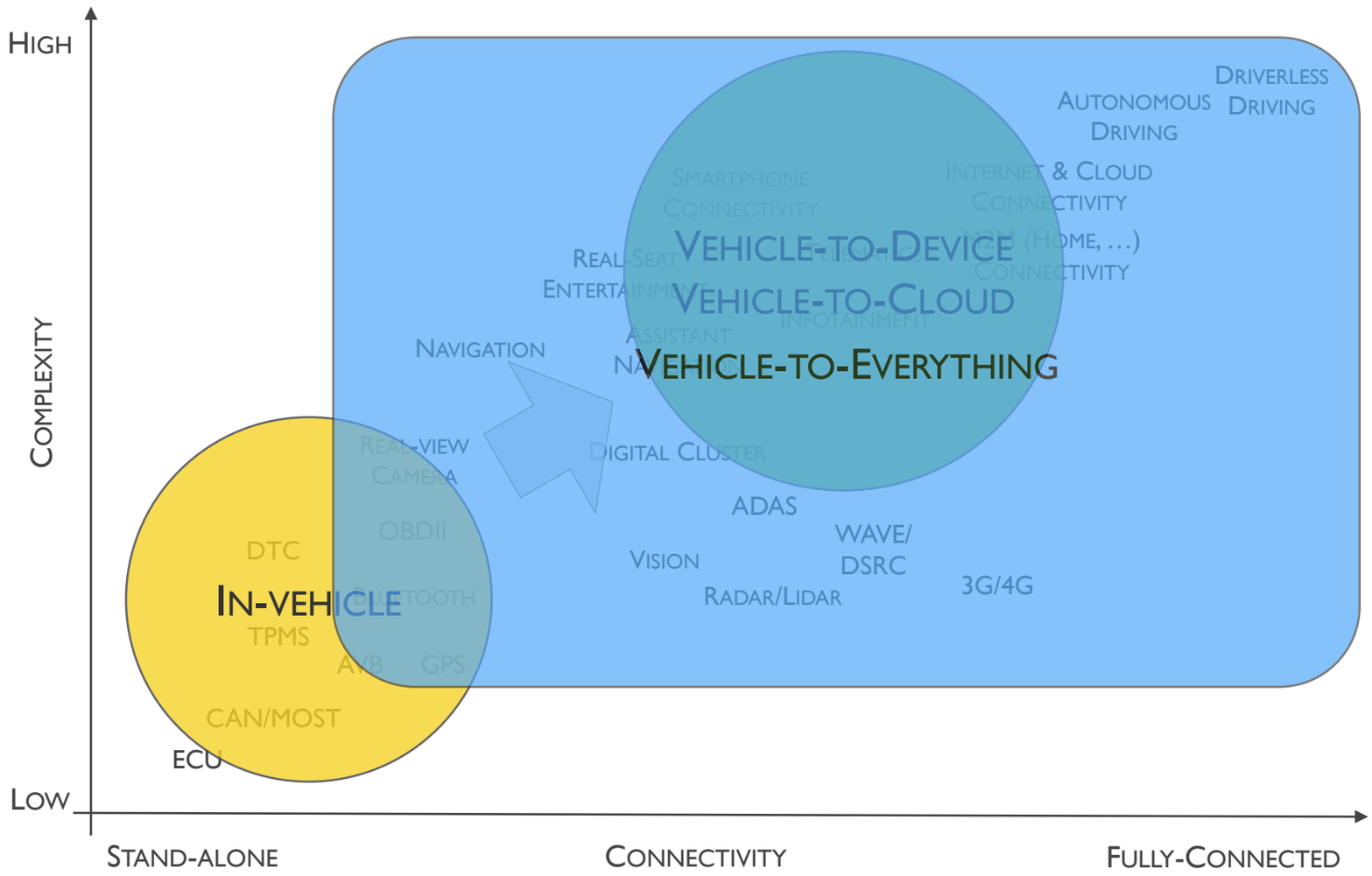
- Global Vehicle-to-X (V2X) modules in new vehicles will reach 62% by 2027
- OEM and aftermarket V2X modules will grow to 423 million by 2027



Source: The Daily Conversation - https://youtu.be/jzSPoDkA1eE

What is V2X Communication and its Future?

- Vehicle to Everything
  - Advanced Driving Assistant System (ADAS)
  - Connected Car → Smart Car
  - Autonomous driving → Driverless driving



V2X COMMUNICATION STANDARDS

❖ WIRELESS ACCESS IN VEHICULAR ENVIRONMENTS (WAVE) STANDARDS

▸ INCLUDING IEEE 802.11P & 1609.X + SAE J2735/2945

STANDARD	USAGE	DESCRIPTION	OSI LAYER
IEEE 802.11-2012 (IEEE 802.11P)	WAVE PHY AND MAC	SPECIFIES THE PHY AND MAC FUNCTIONS REQUIRED OF AN IEEE 802.11 DEVICE TO WORK IN THE RAPIDLY VARYING VEHICULAR ENVIRONMENT	1 AND 2
IEEE 1609.0-2013	ARCHITECTURE	DESCRIBES THE ARCHITECTURE AND SERVICE NECESSARY FOR MULTI-CHANNEL WAVE DEVICES	N/A
IEEE 1609.2-2013	SECURITY SERVICES FOR APPLICATIONS AND MANAGEMENT MESSAGES	COVERS METHODS FOR SECURING WAVE MANAGEMENT MESSAGES AND APPLICATION MESSAGES, IT ALSO DESCRIBES ADMINISTRATIVE FUNCTIONS NECESSARY TO SUPPORT THE CORE SECURITY FUNCTIONS	N/A
IEEE 1609.3-2010	NETWORKING SERVICES	DESCRIBES STANDARD MESSAGES THAT SUPPORT HIGHER LAYER COMMUNICATION STACKS, INCLUDING TCP/IP	2, 3, AND 4
IEEE 1609.4-2010	MULTI-CHANNEL OPERATION	DESCRIBES VARIOUS STANDARD MESSAGE FORMATS FOR DSRC APPLICATIONS AT 5.9 GHZ	2
IEEE 1609.11-2010	OVER-THE-AIR ELECTRONIC PAYMENT DATA EXCHANGE PROTOCOL	DEFINES A BASIC LEVEL OF TECHNICAL INTEROPERABILITY FOR ELECTRONIC PAYMENT EQUIPMENT, I.E. ON BOARD UNIT (OBU) AND ROADSIDE EQUIPMENT (RSE) USING DSRC	7
IEEE 1609.12-2012	IDENTIFIER ALLOCATIONS	INDICATES IDENTIFIER VALUES THAT HAVE BEEN ALLOCATED FOR USE BY WAVE SYSTEMS, INCLUDING THE PROVIDER SERVICE IDENTIFIER ALLOCATIONS HARMONIZED WITH ISO, CEN, AND ETSI.	N/A
SAE J2735-2015	DSRC MESSAGE SET DICTIONARY	COMPRISES A COMPLETE LIST OF ALL DIALOGS (MESSAGES EXCHANGES), MESSAGES, DATA FRAMES (COMPLEX ELEMENTS), AND DATA ELEMENTS (ATOMIC ELEMENTS) WHICH ARE USED IN THE MESSAGE SET	7
SAE J2945 (WIP)	DSRC MINIMUM PERFORMANCE REQUIREMENTS	SPECIFIES THE MINIMUM COMMUNICATION PERFORMANCE REQUIREMENTS OF THE DSRC MESSAGE MESSAGE SETS, THE ASSOCIATED DATA FRAMES AND DATA ELEMENTS DEFINED IN SAE J2735	N/A

SOURCE(MODIFIED): OVERVIEW OF WAVE PROTOCOLS AND STANDARDS, INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY, 2013

IEEE 802.11P & 1609.X

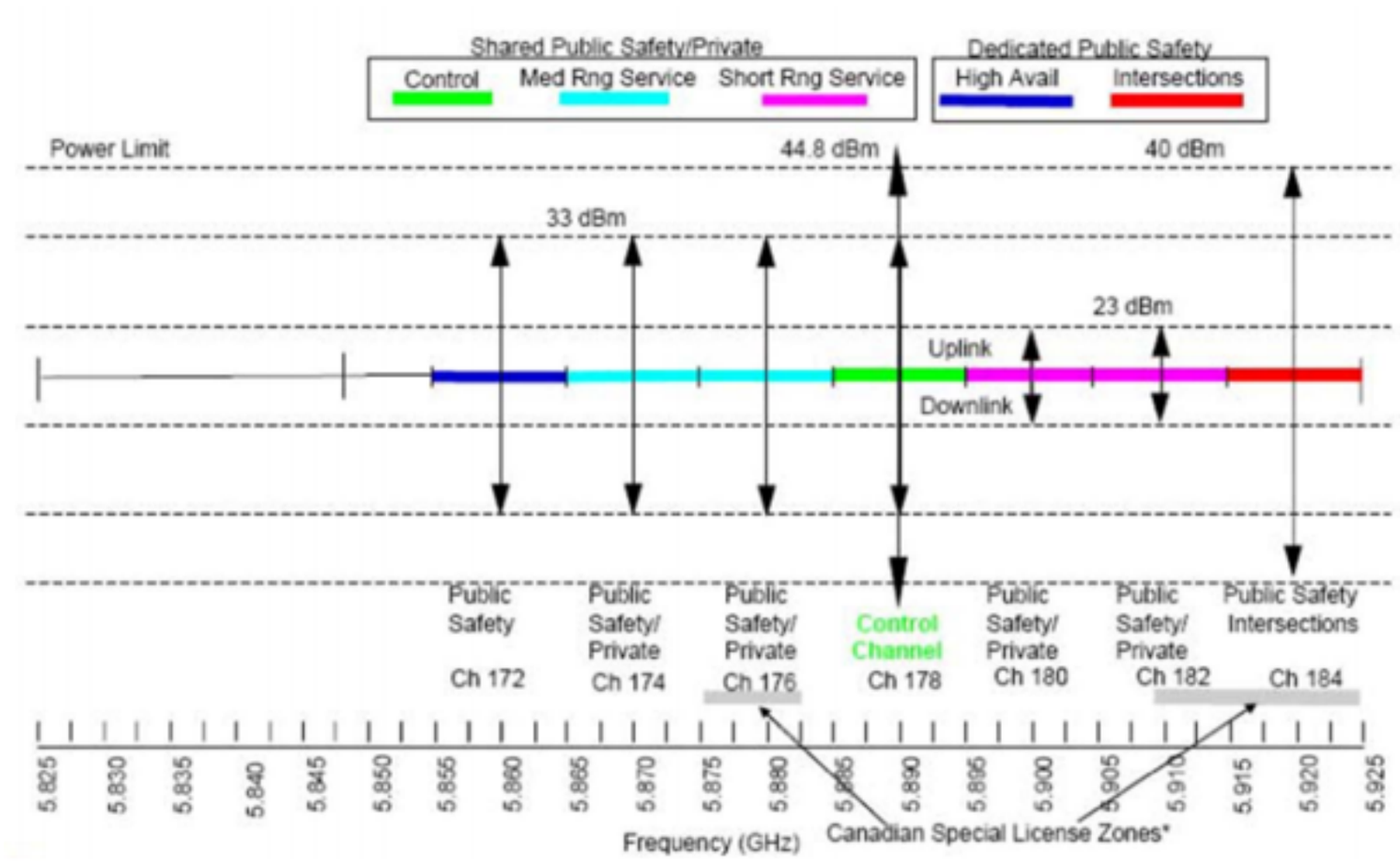
- IEEE 802.11-2012
  - INCORPORATES AMENDMENTS 1 TO 10 PUBLISHED IN 2008 TO 2011
    - IT INCLUDES **IEEE 802.11P-2010**
  - **OUTSIDE THE CONTEXT OF A BSS** (OCB) MODE
    - **DOT11OCBACTIVATED - TRUE OR FALSE**
    - DATA FRAME CAN BE SENT TO EITHER AN **INDIVIDUAL OR A GROUP DESTINATION MAC ADDRESS**
    - **IMMEDIATE COMMUNICATION, AVOIDING LATENCY ASSOCIATED WITH ESTABLISHING A BSS**
    - **NOT UTILIZE IEEE 802.11 AUTHENTICATION, ASSOCIATION, OR DATA CONFIDENTIALITY**
    - **IN A FREQUENCY BAND THAT IS DEDICATED FOR ITS USE**

	JAPAN	US	EU
STANDARD / COMMITTEE	ITS-FORUM	IEEE802.11P/1609.X	CEN/ETSI EN 302 663
FREQUENCY RANGE	755 ~ 765 MHZ	5850 ~ 5925 MHZ	5855 ~ 5925 MHZ
NO. OF CHANNELS	ONE 10 MHZ	SEVEN 10 MHZ (TWO 20 MHZ BY COMBINING 10 MHZ)	SEVEN 10 MHZ
MODULATION	OFDM		
OFDM SUBCARRIERS	3 ~ 18 MBIT/S	3 ~ 27 MBIT/S	3 ~ 27 MBIT/S
OUTPUT POWER	20 DBM (ANTENNA INPUT)	23 ~ 33 DBM (EIRP)	23 ~ 33 DBM (EIRP)
COMMUNICATION	ONE DIRECTION MULTICASTING SERVICE (BROADCAST W/O ACK)	ONE DIRECTION MULTICASTING SERVICE, ONE TO MULTI COMMUNICATION, SIMPLEX COMMUNICATION (BROADCAST W/O ACK, MULTICAST, UNICAST W/ ACK)	
UPPER POROTOCOL	ARIB STD-T109	WAVE (IEEE 1609) / TCP/IP	ETSI EN 302 665 (INCL. GEONETWORKING, …) TCP/UDP/IP

SOURCE - INTELLIGENT TRANSPORTATION SYSTEMS USING IEEE 802.11P, ROHDE & SCHWARZ

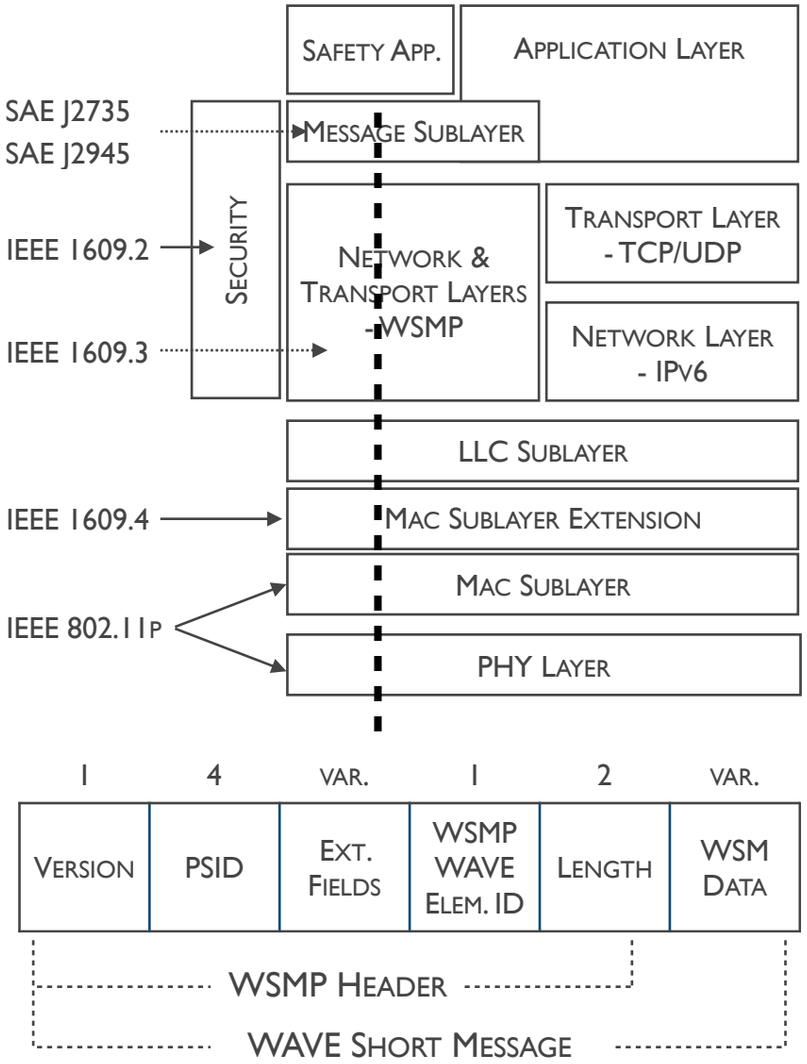
IEEE 802.11P & 1609.X

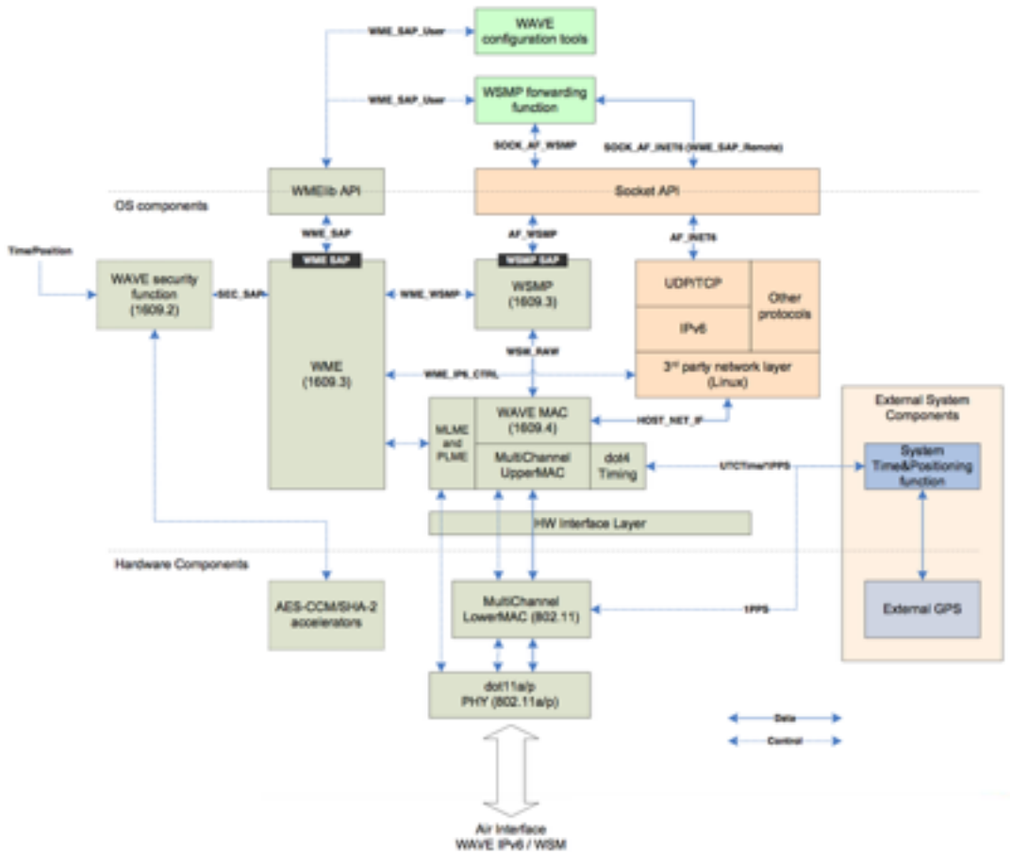
- DSRC SPECTRUM BAND AND CHANNELS IN THE U.S.
  - DATA & MANAGEMENT PLANE SERVICES
  - SAFETY & NON-SAFETY APPLICATION - CONTROL CHANNEL (CCH) / SERVICE CHANNEL (SCH)



IEEE 802.11P & 1609.X

- WAVE SHORT MESSAGE PROTOCOL (WSMP)
  - PROVIDER SERVICE IDENTIFIER (PSID) - SUPPLEMENTARY INFORMATION RELATED TO THE SERVICE





SOURCE: COHDA WIRELESS INTRODUCTION, 2014

SAE J2735

❖ DEDICATED SHORT RANGE COMMUNICATIONS (DSRC) MESSAGE SET DICTIONARY[1)]

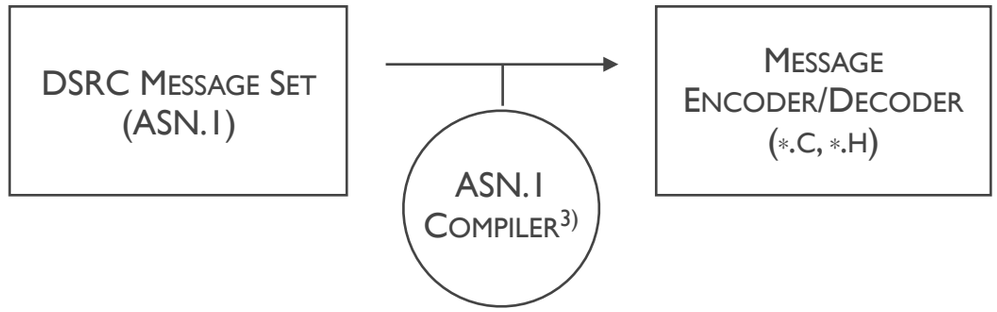
- REVISED AT APR. 2015 (2ND REVISED VERSION AFTER ISSUING AT DEC. 2006)
- SPECIFIED AS ASN.1 REPRESENTATION[2)]
- ENCODING STYLE - BER-DER

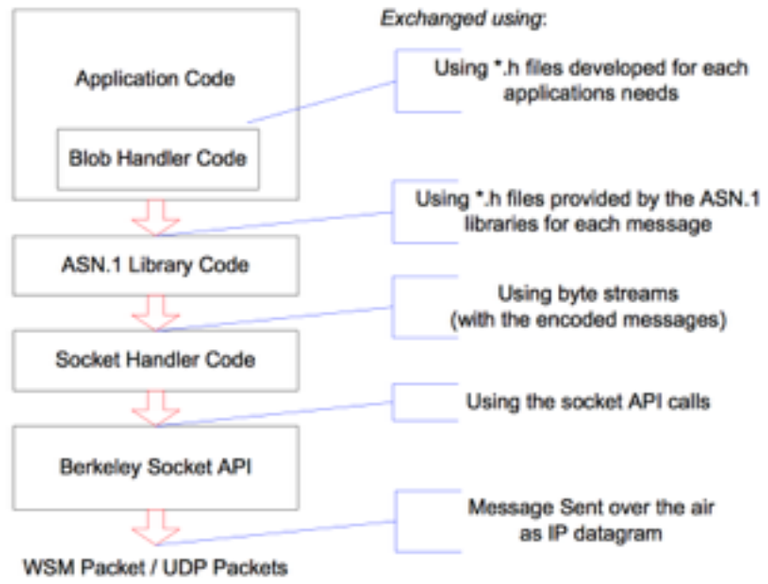
BASIC SAFETY MESSAGE (BSM)
COMMON SAFETY REQUEST (CSR)
EMERGENCY VEHICLE ALERT (EVA)
INTERSECTION COLLISION AVOIDANCE (ICA)
MAP DATA (MAP)
NMEA_CORRECTIONS (NMEA)
PROBE DATA MANAGEMENT (PDM)
PROBE VEHICLE DATA (PVD)
ROAD SIDE ALERT (RSA)
RTCM_CORRECTIONS (RTCM)
…

< SAE J2735 MESSAGE TYPES >

```
BasicSafetyMessage ::= SEQUENCE {
   -- Part I
   msgID       DSRCmsgID,                -- 1 byte

   -- Sent as a single octet blob
   blob1       BSMblob,

   --
   -- The blob consists of the following 38 packed bytes:
   --
   -- msgCnt      MsgCount,              -x- 1 byte
   -- id          TemporaryID,           -x- 4 bytes
   -- secMark     DSecond,               -x- 2 bytes

   -- pos         PositionLocal3D,
     -- lat         Latitude,            -x- 4 bytes
     -- long        Longitude,           -x- 4 bytes
     -- elev        Elevation,           -x- 2 bytes
     -- accuracy    PositionalAccuracy,  -x- 4 bytes
```

< ASN.1 REPRESENTATION OF "BASIC SAFETY MESSAGE" >





< PROTOCOL STACK USED IN THE BSM EXAMPLES >

[1)] SAE J2735 - HTTP://WWW.SAE.ORG/STANDARDSDEV/DSRC
[2)] SAE J2735 ASN.1 SPECIFICATION - HTTP://WWW.SAE.ORG/STANDARDSDEV/DSRC/USA
[3)] ASN.1 COMPILER FOR C/C++ - HTTPS://GITHUB.COM/VLM/ASN1C

Current Status of V2X on Linux

- OCB mode on Linux Mainline Kernel
  - Included in Kernel v3.19
  - Pull request from Wireless-Next 2014-11-07
  - OCB mode supported from Czech Technical University in Prague and Volkswagen Group Research

- OCB mode on IW
  - IW - NL80211 based CLI configuration utility for wireless devices
  - Included in v4.0

MAC80211: 802.11p OCB mode support

This patch adds **802.11p OCB (Outside the Context of a BSS) mode support.**
When **communicating in OCB mode a mandatory wildcard BSSID(48 '1' bits) is used.**
The EDCA parameters handling function was changed to support 802.11p specific values.
The insertion of a newly discovered STAs is done **in the similar way as in the IBSS mode** -- through the deferred insertion.
The OCB mode uses a periodic 'housekeeping task' for expiration of disconnected STAs (in the similar manner as in the MESH mode).
New Kconfig option for verbose OCB debugging outputs is added.

CFG80211: 802.11p OCB mode handling

This patch adds **new iface type (NL80211_IFTYPE_OCB)** representing the **OCB (Outside the Context of a BSS) mode**.
When establishing a connection to the network a **cfg80211_join_ocb** function is called (particular nl80211_command is added as well).
A mandatory parameters during the ocb_join operation are '**center frequency**' and '**channel width (5/10 MHz)**'.
Changes done in mac80211 are minimal possible required to avoid many warnings (warning: enumeration value 'NL80211_IFTYPE_OCB' not handled in switch) during compilation. Full functionality (where needed) is added in the following patch.

IW: Add OCB mode handling

Modify the command for device type setting to support OCB mode. Add **commands for "joining" and "leaving" the OCB mode** network. When joining two arguments are mandatory -- **frequency and channel bandwidth (5 or 10 MHz)**.

RAW SOCKET APPROACH

❖ NEEDS ROOT OR CAP_NET_RAW CAPABILITIES

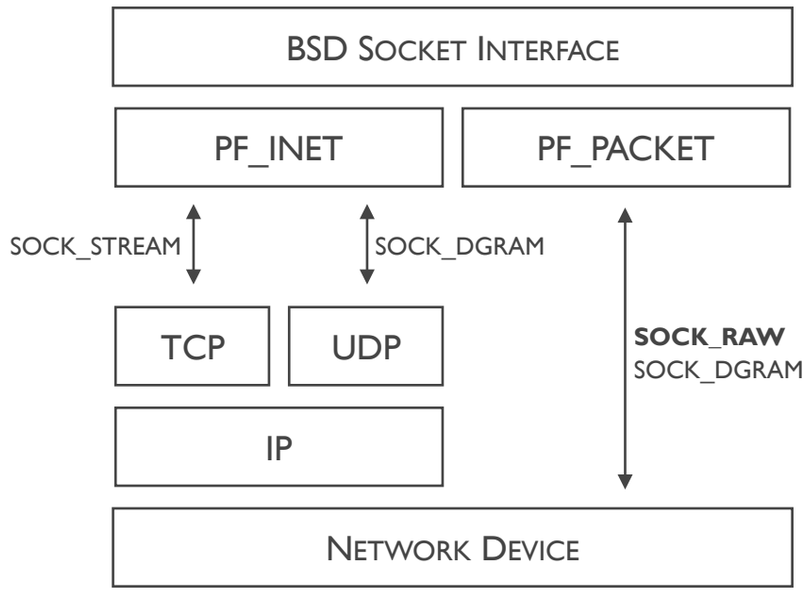
- SET PF_PACKET AS A DOMAIN TYPE
- SET SOCK_RAW AS A SOCKET TYPE
- SET HTONS(ETH_P_WSMP) AS A PROTOCOL TYPE

```
#define ETH_P_WSMP 0x88DC /* WAVE Short Message Protocol */

/* Open PF_PACKET Socket for EtherType ETHER_P_WSMP */
sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_WSMP)))

/* Bind to Device */
setsockopt(sockfd, SOL_SOCKET, SO_BINDTODEVICE, ifName, IFNAMSIZ - 1)

while (1) {
        /* Receive WSMP Packets */
        nread = recvfrom(sockfd, buf, BUF_SIZ, 0, NULL, NULL);
        …
        ber_decode(0, &asn_DEF_BasicSafetyMessage, (void **)&basicSafetyMessage, buf, nread);
        …
}
```

< *RX EXAMPLE* >

```
/* Open PF_PACKET Socket for EtherType ETHER_P_WSMP */
sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_WSMP)))

/* Construct Ethernet Header */
eh = (struct ether_header *)buf;
eh->ether_type = htons(ETH_P_WSMP);
memcpy(eh->ether_shost, src_mac, ETH_ALEN);
memcpy(eh->ether_dhost, dst_mac, ETH_ALEN);
tx_len += sizeof(struct ether_header);

/* Construct sockaddr_ll */
socket_address.sll_ifindex = ifindex;
socket_address.sll_halen = ETH_ALEN;
memset(socket_address.sll_addr, 0xFF, ETH_ALEN);

/* File WSMP packet as payload */
…

while (1) {
        /* Send WSMP packets */
        sendto(sockfd, buf, tx_len, 0, (struct sockaddr *)&socket_address, sizeof(struct sockaddr_ll))
        …
}
```

<*TX EXAMPLE* >



< *PF_INET VS. PF_PACKET* >

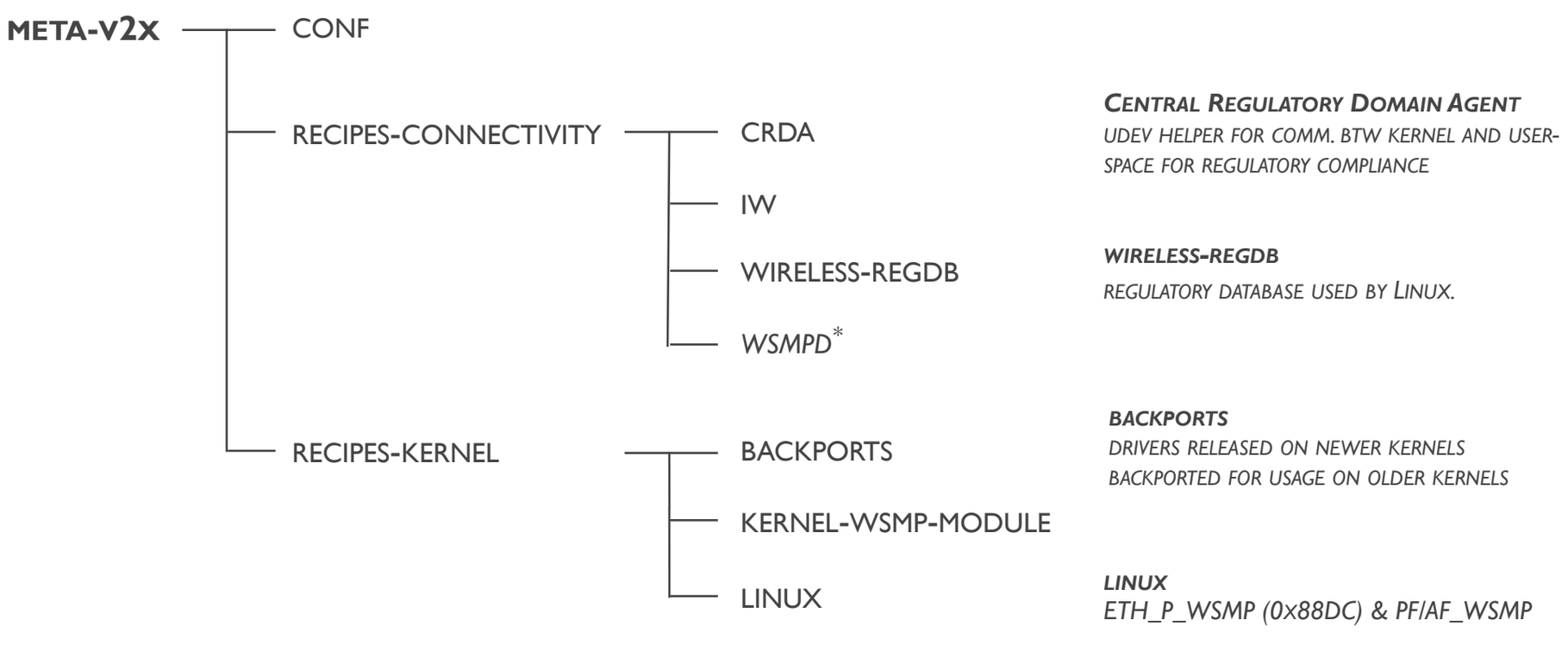
Meta-v2x Yocto Project Layer

❖ Yocto Project

- Open source project for custom embedded Linux-based systems regardless of hardware architecture
- https://www.yoctoproject.org

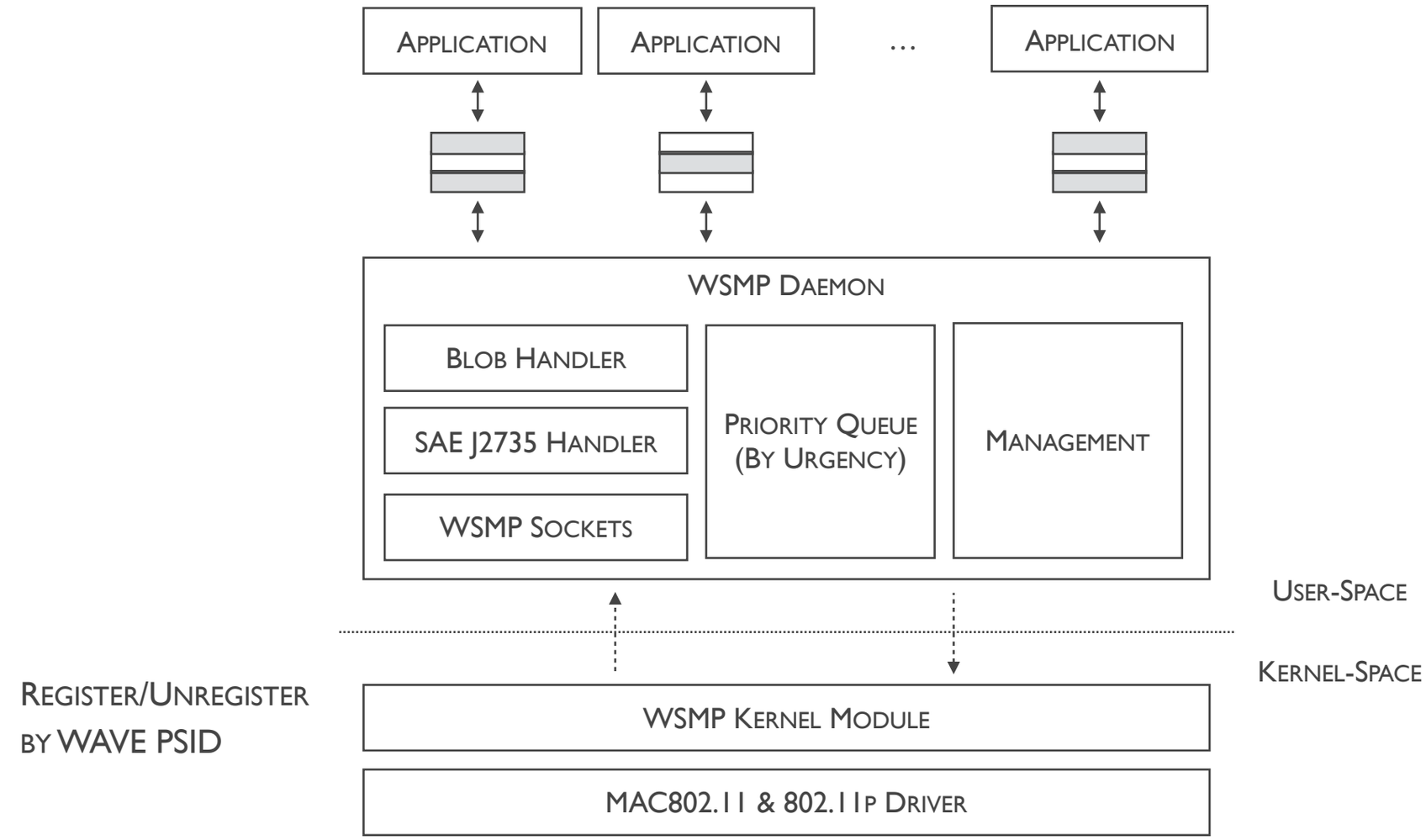
❖ Layer for Vehicle to Everything

- https://github.com/ZENOME/meta-v2x.git
- Based on Yocto v1.8 Fido Release
- Focused on RX-Side
- Work in Progress*



Concept Architecture

- WAVE Short Message Protocol (WSMP) Kernel Module
  - WSMP packet manipulation in Kernel-Space
  - Network + Transport Layer
- WSMP Daemon
  - SAE J2735 Message Handler
  - Multiple Application Support



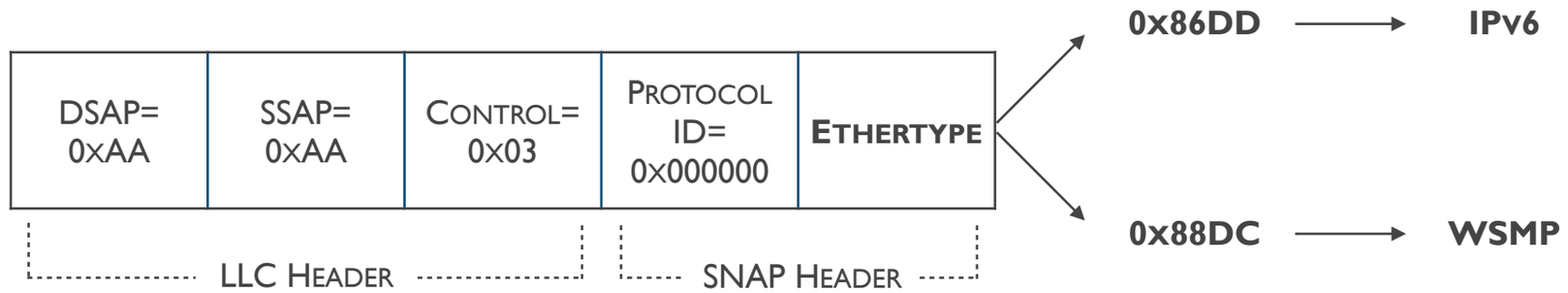
WSMP KERNEL MODULE

❖ 802.11P DRIVER

- QUALCOMM ATHEROS 5GHZ 802.11A MINI PCI-E CARDS
  - AR9592/AR9462 - HALF CHANNEL WIDTH (10 MHZ)
  - "ATH9K: ADD OCB MODE SUPPORT" PATCH BY ROSTISLAV

❖ WSMP

- ADD WSMP ETHERNET TYPE AS "0X88DC"
- ADD WSMP SOCKET & PROTOCOL FAMILY HANDLER




```
#define ETH_P_WSMP 0x88DC /* WAVE Short Message Protocol */

static struct packet_type wsmp_packet_type __read_mostly = {
        .type = cpu_to_be16(ETH_P_WSMP),
        .func = wsmp_rcv,
};

static int __init wsmp_init(void)
{
        …
        dev_add_pack(&wsmp_packet_type);
        return 0;
}
```

< ADD WSMP ETHERNET TYPE >

```
static struct proto wsmp_proto = {
        .name           = "WSMP",
        .owner          = THIS_MODULE,
        .obj_size       = sizeof(struct wsmp_sock),
};

static const struct proto_ops wsmp_proto_ops = {
        .family         = PF_WSMP,
        .owner          = THIS_MODULE,
        .release        = wsmp_release,
        …
        .recvmsg        = wsmp_recvmsg,
        …
};
```

< ADD WSMP SOCKET & PROTOCOL FAMILY HANDLER >

WSMP Socket

❖ Simple Usage

- Similar as a UDP Socket

❖ Protocol Specifics

- Set PF_WSMP as a Domain Type
- Bind by Provider Service Identifier (PSID)
- Needs a lot of improvements

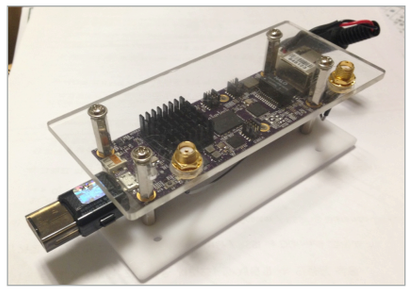
```
/* WSMP Socket Address */
struct sockaddr_wsmp {
        __kernel_sa_family_t wsmp_family;
        unsigned int wsmp_psid;
};
```

```
/* WSMP Socket Address */
struct sockaddr_wsmp addr;

/* Create PF_WSMP Socket */
sockfd = socket(PF_WSMP, SOCK_DGRAM, 0)

/* Set WSMP Socket Address */
memset(&addr, 0, sizeof(struct sockaddr_wsmp));
addr.wsmp_family = AF_WSMP;
addr.wsmp_psid = 0x0;

/* Bind by PSID */
bind(sockfd, (struct sockaddr *)&addr, sizeof(struct sockaddr_wsmp))

while (1) {
        /* Receive WSMP Packets */
        nread = recvfrom(sockfd, buf, BUF_SIZ, 0, NULL, NULL);
        …
}
```

< WSMP Socket - RX Example >

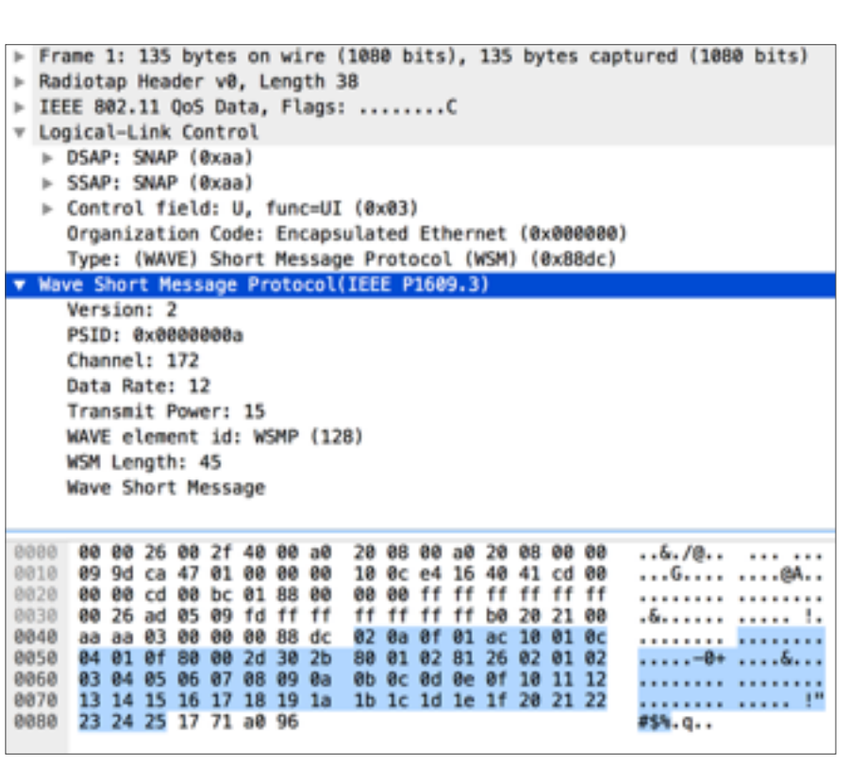
COMM. TEST W/ COMMERCIAL VENDORS

- SINGLE BOARD COMPUTERS W/ MINI PCI-E SOCKET
  - FREESCALE™ I.MX6 800MHZ DUAL LITE
  - W/ AR9592 MINI PCI-E CARD



```
# Set wireless regulation as US
iw reg set US

# Set wlan0 as OCB mode & join to 5860 channel
iw dev wlan0 set type ocb
ip link set wlan0 up
iw dev wlan0 ocb join 5860 10MHZ

# Add Radio-Tap monitoring interface
iw dev wlan0 interface add mon0 type monitor flags none
ip link set mon0 up

# capture WSMP packets
tcpdump -i mon0 -w wsmp.pcap ether proto 0x88dc
```

< WSMP PACKET CAPTURE PROCEDURE >



< CAPTURED WSMP PACKET IN WIRESHARK >

THANK YOU

zenome

CONNECTED DRIVING FOR LIFE