Hybrid Network Architecture for Inter-ECU Communication

by

Wei Li

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering
(Computer Engineering)
in the University of Michigan-Dearborn
2019

Acknowledgments

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank General Motors Corporation (GM) for supporting me to do the necessary research work and giving and confirming the permission to encouraged me to go ahead with my thesis.

I am deeply indebted to my thesis committee members, who have imparted invaluable academic and professional wisdom.

Dr. Samir Rawashdeh is my thesis advisor. His help, stimulating suggestions and encouragement helped me in all the time of researching for and writing of this thesis. He provided me thoughtful input and clear direction throughout the work.

Dr. Weidong Xiang encourage me all along, generously sharing his expertise and insight in network communications. He also taught me to think more critically and insightfully.

Dr. Selim Awad has shared not only academic advice but also tips on my careers. I admire his great personality and professionalism.

Besides, I would like to thank my peers and friends, Zaid El-Shair and Sheng Liu, for the creative discussions, for the sleepless nights we were working together before deadlines, and for all the funs we have had in the last few years.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Table of Contents

List of Tables

List of Figures

Abstract

With the growth of the number of automotive electronics and the development of autonomous driving technology, the demand for high-bandwidth of in-vehicle networks increases significantly. Traditional technologies such as CAN and FlexRay are facing the challenge of bandwidth limitations. For example, cameras loaded on the side and back of the vehicle are used to detect objects in the blind spots, which needs a lot of bandwidth to transmit images and videos to the ECU. Traditional wired vehicular networks cannot afford such a large amount of data transmission. This is reasonable because it is difficult to predict the bandwidth demand of today's automotive network at the time when traditional technologies are developed. To improve network communication performance, current vehicle network architectures use a combination of several different network protocols to meet the growing bandwidth requirements. Also, for communication reliability, many network protocols working together form a redundancy mechanism. An obvious example is that wireless networks can act as channel backups to carry on important data transmissions when critical transmission failures happen on wired networks.

Beginning with this point, the thesis studies a hybrid network communication within vehicle ECUs. This work is basically divided into four parts. First, it introduces the background and related works. Second, a prototype of hybrid network architecture is demonstrated. It is developed for in-vehicle communications using several developed network protocols, including Ethernet, dual-band Wi-Fi, CAN and BLE. Four experimental results of single-interface benchmarking tests are carried out. Next, a preliminary hybrid network algorithm is carried out, which aims to improve the whole system performance in terms of data transmission reliability and bandwidth utilization. Redundancy transmissions are the basis of this algorithm. Finally, it presents several fault detection experiments which are used for developing detection rules. Based on these rules and the algorithm which is mentioned above, a preliminary recovery mechanism is developed and verified by using testbed, to simulate how the system performs when critical transmission failure happens.

Chapter 1 Introduction

1.1 Background

In recent years, the number of electronic control units (ECUs) in vehicles has increased significantly. Control area network (CAN) and high-level device profiling protocols, used as stand-alone fieldbus system for vehicular embedded solutions to inter-ECU communication, have exposed the shortcomings of fulfilling both bandwidth and reliability requirements in future automated vehicles. Applications which are developed based on intra-vehicle communication, such as GPS navigation, sensors for speed and tire pressure and audio or video systems, are highly dependent on the reliability of inter-ECU network transmissions. As the number of sensors increasing, massive data transmissions can bring many challenges, including safety issues, background traffic influences, wireless extensions and so on.

A Large number of wired sensors and wireless sensors forms a complicated hybrid network carrying a lot of data transmissions, which is a great challenge in vehicular networks [1]. In previous vehicle network architecture, each electronic application was implemented by adding a new stand-alone ECU. This is not an efficient way when the numbers of sensor increased because it led to networks growing fast in complexity. To overcome this problem, the bus-based networks such as CAN and FlexRay were invented, which are introduced in more detail in related work. Multiple ECUs on the bus-based networks are connected to each other, which is an improvement on the point-to-point links. It optimized data transmissions by using different topologies under different situations. However, the bandwidth consumed significantly increases as the number of devices on bus-based networks grows.

Besides the bandwidth limitation of CAN and traditional wired networks, in-vehicle networks also have performance degradation from wireless sub-networks in terms of communication reliability [2]. Wireless networks are not as reliable as wired networks. Wireless transmissions are easily degraded by interferences from other signal sources from in-vehicle networks or the environment around the vehicle. Current in-vehicle wireless applications are more about non-time-critical tasks such as the wireless key and Bluetooth for mobile phones. As a development trend of network structure, wireless approaches have many advantages in sensor networks. Wireless devices are easy to be deployed without constrained by position. The whole weight of vehicles would decrease with fewer cables, which is good for fuel saving. Wireless techniques have been developed for years to meet different requirements such as high transmission rate or low energy consumption, which means there are many options when applied in vehicle applications. Wireless techniques have potentially capability to carry on more critical data transmissions in real-time in-vehicle network communications. They could be used to share the bandwidth load from the wired network or to be as a backup of important data transmissions just in case of wired transmissions fail. Wireless approaches are introduced in more detail in related work.

This paper aims to propose an in-vehicle hybrid network architecture for inter-ECU communications and some experimental result analysis with prototype testbed. The current in-vehicle network techniques and protocols are introduced in the rest part of Chapter 1. A Prototype of the hybrid network is demonstrated in Chapter 2 with architecture design and explanations of experimental parameters. Chapter 4 explores the hybrid network algorithm which is a priority-based decision-making mechanism. It is a preliminary optimization of bandwidth usage in terms of critical data transmission. Chapter 3 and 5 are experiments of no-fault benchmarking and fault detection, respectively. Chapter 6 shows how the entire hybrid network system works while the algorithm is enabled. Finally, Chapter 7 contains conclusions and the future directions of work in this area.

1.2 Related Work

1.2.1 Wired Approaches

Vehicular networks enable many applications associated with driving safety, traffic efficiency and infotainment. Hundreds of devices in a vehicular network have different requirements of the network performance. The modern in-vehicle device domains can be divided into four parts: powertrain, chassis, bodywork and comfort and driver assistance [3]. The powertrain consists of components which provide power to the vehicle, including engine, axes, and wheels. Their sensors could measure pressure, speed, and stability. The controller must sample parameters with a high frequency. Chassis includes breaks, steering system and suspension, which has the same time restriction as the powertrain. The bodywork and comfort have elements such as air conditioning, heating, window, and door locks. These sensors and controllers require a small amount of bandwidth. High latency is acceptable in this domain. Driver assistance, such as GPS and automatic parking, has different requirements of the bandwidth under different sensors.

A general overview of current research on intra-vehicle networks is presented in [1]. It introduces techniques about vehicular network architecture in three parts: physical layer with devices, link layer protocol, and middleware with applications.

In the physical layer, many technologies have been developed and applied in practice for years, such as Control Area Network (CAN), FlexRay, Local Interconnect Network (LIN), Media Oriented Systems Transport (MOST) [4], Low-Voltage Differential Signaling (LVDS) [5] and IEEE 1394 Firewire [6]. People use different technologies to meet different requirements in the network, such as bandwidth, data rate, and economic factors. Currently, in the complex in-vehicle networks, automotive network traffic can be divided into several categories: control data, safety data, infotainment data, and driver assistance cameras. In control data, some data are from low-bandwidth control applications demanding low bandwidth and low QoS, which is transmitted through LIN and CAN [7]. Some are from real-time control applications which utilize CAN and FlexRay, as well as the safety data. These data require low bandwidth and high QoS. The infotainment data and Driver assistance cameras are mainly transmitted through LVDS which has high bandwidth and high QoS [8].

Link layer protocol in the automotive domain mainly includes three techniques: IEEE 802.1Q [9], Audio Video Bridging (AVB) Ethernet [10], and Time-Triggered Ethernet (TTEthernet, AS6802 [11]). Ethernet itself is not configured to provide safety-critical functionality by default, which is necessary for the automotive domain. These Ethernet approaches mentioned above are found to overcome this problem. IEEE 802.1Q allows virtual LANs (VLANs) to priority tag packets by adding an extra field of a priority value to the Ethernet header of a packet. AVB is a technology which consists of four IEEE standards: IEEE 802.1AS, IEEE 802.1Qat, IEEE 802.1Qav, and IEEE 802.1BA. It is mainly designed for streaming audio and video sources simultaneously over Ethernet. TTEthernet is designed to enable the coexistence of time-triggered real-time synchronized communication with lower priority event-triggered frames over Ethernet. One of its advantages is that lower priority messages are interrupted to enable time-triggered messages to get priority. This feature is also one of the investigations in the second revision of the AVB Ethernet [12].

A performance comparison of IEEE 802.1Q and AVB is presented in [13]. A network simulation tool named OMNeT++ with the INET-framework is used to make an analysis of network performance. The result shows that AVB improves around 40% of the maximum end-to-end delay over two hops compared with IEEE 802.1Q on the driver assistance streaming frames. However, the control frames in AVB have worse performance than the IEEE 802.1Q because of the lower priority values.

In [14], a demonstration that background cross-traffic of real-time Ethernet protocols can have a significant impact on in-vehicle backbone networks is presented. The author compares real-time approaches Ethernet AVBs asynchronous credit-based shaping with the time-triggered and rate constrained traffic classes of TTEthernet. The result of their simulation is that cross-traffic may increase end-to-end latency by more than 500% while the jitter can become 14 times higher than for a network without background tasks.

An evaluation of real-time approaches for the in-vehicle network is given in [15] by comparing TTEthernet with FlexRay. The work shows some comparisons on latency, jitter, and bandwidth. For latency and jitter, they have comparable results. For bandwidth, as the payload size increases, the TTEthernet has more available real-time slots left than the FlexRay. Thus, the author notes that it is possible to transfer a FlexRay in-vehicle network system to a TTEthernet-based system.

With modern advances in driver assistance and safety technology in vehicles, Ethernet is the obvious choice as the network backbone to support these high bandwidth applications. It is fast, inexpensive, flexible, and ubiquitous. The work presented in [16] is a novel automotive network simulation platform which uses a testbed approach to integrate real video streams. Testing is carried out on two candidate automotive Ethernet topologies, namely Double Star and Daisy Chain. The author demonstrates that under severe load. In this case, six in-vehicle cameras broadcasting simultaneously, along with control, infotainment and miscellaneous traffic, delay requirements for safety-critical video information may be violated. Ethernet will likely act as a high-speed backbone network with legacy technologies in automotive networks.

AUtomotive Open System ARchitecture (AUTOSAR) [17] is an industry solution of middleware. It aims to define a unified scalable interface between applications and automotive ECUs to enable more rapid and generalized development. Based on AUTOSAR, applications can be developed once and installed multiple times, thus getting more development efficiency and less complexity. A work in [18] presents a framework to analyze automotive networks in AUTOSAR which is composed of FlexRay, CAN and LIN. The result shows that AUTOSAR can increase scheduling resource utilization. Another work in [19] shows scheduling analysis in AUTOSAR. Four basic features of an analyzable model are presented in this paper: application workload, timing behavior, resource platform, and allocation. The author measures how the AUTOSAR system model matches these features.

1.2.2 Wireless Approaches

Wire technologies have been developed for years in the automotive system. As the number of sensors is increasing, cables will bring significant weight issue to the vehicle mass. Also, the installation and maintenance of sensors on cables are inconvenient. Thus, the wireless approach of interconnection among sensors and ECUs can be used to reduce the complexity of intra-vehicle networks. Some exit wireless technologies are listed in [20]: Bluetooth, ZigBee, Radio-Frequency Identification (RFID), Ultra-Wideband (UWB), and 60 GHz Millimeter-Wave (MMW). Features of these wireless technologies are shown in Table 1.

Wireless Solution	Bluetooth	ZigBee	RFID (Passive)	UWB	60 GHz MMW
Frequency	2.4 GHz	868/915 MHz and 2.4 GHz	915 MHz	3.1-10.6 GHz	57-64 GHz
Data Rate	1, 2, 3 Mb/s	20-250 kb/s	<4Mb/s	53.3-480 Mb/s	>1 Gb/s
TX Power	1, 2.5, 100 mw	<1 mw	0	1 mw/Mb/s	10 mw
Transmission Range	<10 m	<100 m	<10 m	<100 m	Short Range
MAC Protocol	TDMA	CSMA/CA	EPC global	CSMA/CA and TDMA	CSMA/CA and TDMA
Modulation	GFSK/DQPSK/8DPSK	BPSK/O-QPSK	BPSK	MB-OFDM	Single carrier OFDM

Table 1. Vehicular Network Wireless Approaches

Different wireless technologies focus on different applications. Bluetooth Class technologies are considered as a replacement of cables on the vehicle. It is mainly used in the multimedia systems on the vehicle. However, A Bluetooth network can only support eight active devices. Its poor scalability limits Bluetooth network to be expanded to a larger WSN. ZigBee is based on IEEE 802.15.4 which is a standard of low-rate wireless personal area networks (LR-WPANs). It can be applied on monitoring and control networks, because of its flexible scalability and large transmission range. An analysis of transmission latency over ZigBee is conducted in [21]. The Passive RFID is to use an RFIP tag attached to every sensor and a reader connected to the ECU to receive data from sensors periodically. Low cost and no power supply allow RFID tags to be easily deployed, but it has a challenge that critical data transmission is hardly guaranteed when collisions and interference happens. Reliable MAC protocols [22] should be developed to addressing this challenge for large-scale RFID networks. UWB is a kind of radio technology operated in a range of 3.1-10.6 GHz frequency band. Due to its resistance to severe wireless channel fading and shadowing and high time domain resolution, UWB is suitable to be used on localization and tracking. Research based on the testbed designed in [23] shows that UWB is highly reliable when it is used to transmit automotive speed data from four-wheel speed sensors to the ECU. MMW has a much higher data rate than other wireless technologies. An in-vehicle MMW channel model presents in [24]. A measurement of the channel impulse response (CIR) and scale fading on 55-65 GHz is carried out as well.

A performance evaluation work of Bluetooth Low Energy (BLE) for V2V communication is presented in [25]. As a relatively new technology, BLE has been adopted by many kinds of intelligent electronic devices, such as smartphones and smartwatches. In BLE, the role of client or server is defined by Generic Attribute Profiles (GATT). Every profile can define one or more services to specify several attributes or characteristics. The operation modes of a BLE device can be Peripheral, Central, Broadcast, and Observer. In the paper, the experiments are under the Peripheral/Central modes where server devices (Peripherals) provide data to client devices (Centrals) that subscribe to Peripherals. The result shows that the round-trip time (RTT) varies from 0.06 s to 0.12 s and the received power decrease from -75 dBm to -92 dBm when the distance of two static devices increases from 0 m to 100 m.

The network topology is the arrangement of devices or links. In general, topologies of the intra-vehicle network have several types: star, daisy chain, tree, ring, double star, and point to point. In wired approaches, as has been discussed in [1], some related works with different topologies are introduced and analyzed with using Ethernet techniques including 802.1Q, AVB and TTEthernet. In wireless approaches, Bluetooth and RFID use a star topology, which is fixed by protocols. Zigbee and UWB have more flexible topologies such as the star, mesh, and hybrid. However, most works are implemented based on simulations, and only a few works have prototypes by using topologies of point-to-point and daisy chain. Thus, it is worthy to develop a hybrid intra-vehicle network prototype with multiple network techniques by using double-star topology.

Chapter 2 Hardware Testbed

In this chapter, a prototype design of the inter-ECU hybrid network is carried out. It is demonstrated in the following sections: an overview of the architecture, network interfaces, hardware, software, and hybrid network algorithm design.

2.1 Architecture Design

The architecture of the prototype and the testbed are shown in Figure 1 and Figure 2. With using double-star topology, it has multiple processing units and relative communication techniques. There are two kinds of processing units, Electronic Control Unit (ECU) and Device Control Unit (DCU). ECUs are primary controllers for subsystems such as engine and transmission. DCUs are lower level controllers that interface to sensors and actuators and exchange data with ECUs. There are two segments of the network architecture: backbone networks and device subnets. The backbone network interconnects two ECUs through Ethernet and Wi-Fi. The device subnet interconnects two DCUs with one ECU over CAN and BLE. Considering the requirement of configurable features on Wi-Fi in Phase II, the router is separated by an Ethernet switch and a custom Wi-Fi router. Device subnets are connected to each other by both wired and wireless interfaces as primary and secondary interfaces, respectively. All processes are monitored on the desktop by using SSH and CAN sniffer.

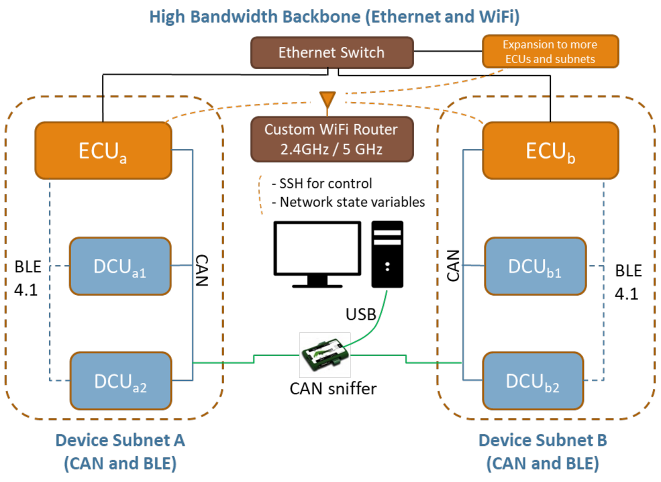

High Bandwidth Backbone (Ethernet and WiFi)
Ethernet Switch
Expansion to more ECUs and subnets
Custom WiFi Router 2.4GHz / 5 GHz
ECU_a
ECU_b
- SSH for control
- Network state variables
BLE 4.1
CAN
DCU_{a1}
DCU_{a2}
DCU_{b1}
DCU_{b2}
USB
CAN sniffer
Device Subnet A (CAN and BLE)
Device Subnet B (CAN and BLE)

Figure 1. Hybrid Network Architecture

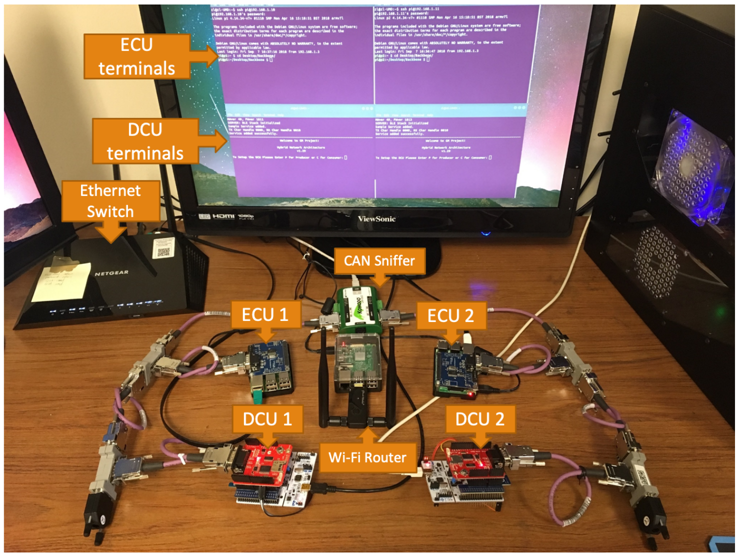


Figure 2. The testbed of hybrid network. In the following experiments, only one DCU are constantly enabled in each device subnet. Other two DUCs are ready to be added into the system whenever they are needed.

2.2 Network Protocols

The prototype includes four different network interfaces, including Ethernet, Wi-Fi, CAN, and BLE. Ethernet and Wi-Fi form backbone networks which are used in ECU to ECU communications. CAN and BLE forms device subnets which are used in DCU to ECU communications. Wired interfaces would set as primary networks and wireless interfaces are secondary networks. The data transmission takes priority of primary networks, and secondary networks would act as a transmission backup to in case failures happen on primary networks.

Primary Backbone Network: The primary backbone network is based on the Ethernet protocol (IEEE 802.3). Current setup can support both 100Base-T and 1000Base-T with data rates of 100 Mbps and 1 Gbps, respectively. The maximum payload for each protocol is 1500 bytes, while the minimum packet size for 100Base-T and 1000Base-T are 64 bytes and 512 bytes, respectively. Ethernet provides tremendous link capacity with low latency on a single segment and should be able to support most vehicular data traffics.

Secondary Backbone Network: The secondary backbone protocol is Wi-Fi (IEEE 802.11), while our devices support 802.11 b/g/n/ac variants of the protocol. Wi-Fi is a wireless protocol that operates at 2.4 GHz or 5 GHz ISM frequency bands. The 2.4 GHz band has better propagation characteristics compared to 5 GHz but also suffers from more interference. All variants use OFDM modulation scheme with nominal bitrates adjustable from 6 Mbps to 72 Mbps. Effective bit rates in a vehicle are expected to be substantially lower than the nominal data rate in order to achieve acceptable reliability. In the prototype, the data rate can be configurable on a customized Wi-Fi router. In the following experiments with using 802.11n, bitrate of 2.4 GHz and 5 GHz bands are both 65 Mbps with a 20 MHz bandwidth and bitrate of 5 GHz band with a 40 MHz bandwidth is 90 Mbps, normally.

Primary Device Subnet: The primacy device network is the controller area network (CAN) using extended frames (ISO 11898), also referred to as CAN 2.0B. CAN is a priority arbitrated network designed for device control. As such, the payload sizes range from 8 bytes to 0 bytes. The overhead in a CAN frame is 64 bits, leading to a maximum frame length of 128 bits (not including bit stuffing). For vehicle length networks (less than 40m) the maximum CAN data rate is 1 Mbps, while the actual data rate used in the prototype is 500 kbps.

Secondary Device Subnet: The secondary device network is Bluetooth Low Energy (BLE). Bluetooth is a wireless protocol operating in the 2.4 GHz ISM band with 1 Mbps data rate and 10 dBm maximal transmission power. Bluetooth 4.1 is used in the prototype. Besides BLE features, Bluetooth 4.1 improves data transmission rate and supports devices to has multiple roles simultaneously compared to Bluetooth 4.0. The communication range is determined by both channel conditions and transmission power. BLE uses frequency hopping and forwards error control to improve reliability.

2.3 Hardware and Software Design

The ECU is prototyped by Raspberry Pi 3 B+ with PiCAN2. The DCU is prototyped by STM32 F411RE microcontroller with off-board CAN/BLE interfaces. The customized dual-band Wi-Fi router is carried out based on Raspberry Pi 3 B (RPi3B) with a dual-band Wi-Fi adapter. Programs on ECU, DCU and the customized Wi-Fi router are developed by using C. In backbone networks, transmission frames can be divided into three types: probe frame, data frame, and background traffic frame. A specialized frame structure is shared within all three data types, which consists of devices address and time stamp and other useful information for routing. However, values in frame structure vary among different data types. In device subnets, types of transmission frames include a probe frame and a data frame. The frame structures are different depending on the standards of CAN and BLE. Probe frame, data frame, and background traffic frame are introduced below.

Probe frames are used to estimate channel state as system running, including round-trip latency (RTL), packet drop rate (PDR), throughput, and jitter. It is also treated as a part of the network state estimator in hybrid network algorithm design which is mentioned in Section 2.5. It is not only used in backbone networks but also device subnets. The size of ECU probe frames is 28 bytes. It only takes a hybrid network algorithm head but without any payload. On DCUs, the size of probe frames is different from CAN to BLE. For CAN, it is 108 bits, which includes 44 bits of CAN standard head and end and 8 bytes of payload. For BLE, it is 18 bytes, which consists of 1 byte of the preamble, 4 bytes of access address, 2 bytes of data unit head, 8 bytes of payload and 3 bytes of CRC.

Data frames are used to transmit the actual data which ECUs receive from DCUs. The size of the data frames is 36 bytes. It consists of the same head as ones in probe frame and 8-byte payload. In the payload, the first byte is reserved for routing information. It consists of 3 bits of source address, 3 bits of the destination address and 2 bits of classes. That means it has a capability to configure backbone networks of up to 8 ECUs and 8 DCUs on each device subnet. Data frames have three classes to simulate different priorities on safety critical messages, drivetrain, and infotainment.

Background traffic frames are generated by ECUs. They are mainly created for experiments. The size of the background traffic frame can be changed from 28 bytes to 1400 bytes by configuration, and its transmission throughput can be configured as well.

Network status measurements are explained as follows:

Round-trip Latency (RTL): RTL measures the length of time it takes for a packet to be send from a source device to a destination device and back again. It includes the processing time on destination. It is measured on Ethernet, Wi-Fi, CAN and BLE. The general procedure of latency measurement is shown in Figure 3. When the packet is generated on the Device A, a time stamp t_1 is recorded. Then it is transmitted to the Device B, which will echo back as soon as possible. Finally, when Device A receives echo the time stamp t_2 is created. The RTL of packet n is:

$$RTL_n = t_{n2} - t_{n1}$$

The overall average of round-trip latency is:

$$\overline{RTL} = \frac{1}{n} \sum_{i=1}^{n} RTL_i$$

For backbone networks, Ethernet and Wi-Fi, Device A and Device B are ECU$_a$ and ECU$_b$, respectively. For subnet networks, CAN and BLE, Device A, and Device B are the ECU and the DCU, respectively. If packet n is considered as a dropped packet, then $RTL_n = 0$. It is not counted in $\overline{RTL}$. RTL is measured on the user level.

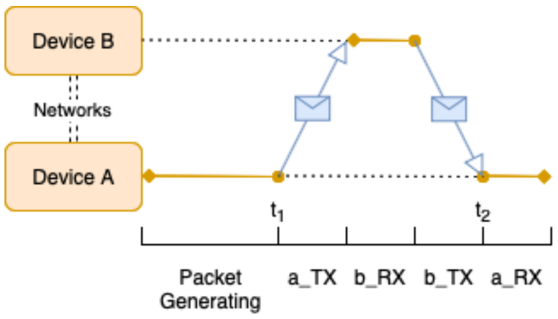


Figure 3. Round-trip Latency Measurement

Packet Drop Rate (*PDR*): PDR is mainly measured in Ethernet and Wi-Fi. In the prototype, the dropped packet is not only defined by dropping in the physical layer but also depending on RTL deadline which is manually configured in software. That is, if RTL_i is larger than the deadline, the packet i would be treated as a packet drop. The reason for this configuration is to ensure the timeliness of the data transmission. In backbone networks, Ethernet and Wi-Fi, all RTL deadlines for PDR are set to 80% of the transmission period of probe frames in the following experiments. For example, the period of probe frames transmission is configured as 10 ms, and the deadline for dropped packets should be set to 8 ms. That is, if the RTL of a probe frame is larger than 8 ms, it will be considered as dropped because it is not effective any more. In CAN and BLE, there is no RTL deadline and PDRs for both only depend on transmission state on the physical layer.

Throughput with ACKs: It refers to how many bits of packets are successfully transmitted per second. It is working with ACKs from the destination devices. In the prototype, it can be represented as below:

$$T = \frac{N_p}{t} \times L(p) \times 8$$

N_p is the number of packets already successfully transmitted with echoing received and the $L(p)$ is the size of the packet in bytes. t is the current running time in a microsecond.

Jitter: Jitter is defined as the latency deviation. It measures variability over time of the RTL across a network. In the prototype, current jitter J_i is calculated based on latencies of the last 100 transmitted packets. It can be represented as:

$$J_n = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(RTL_i - \overline{RTL})^2} \qquad (n < 100)$$

and:

$$J_n = \sqrt{\frac{1}{100}\sum_{i=n-99}^{n}(RTL_i - \overline{RTL})^2} \qquad (n \geq 100)$$

Receive Signal Strength Indicator (RSSI): RSSI indicates the signal strength of a receiving device. Signal strength is usually invisible, but it affects functionality in a wireless network. It is an important parameter to measure the wireless environment. In the prototype, the data of RSSI is directly extracted from parameters of the wireless interface of ECUs.

All the devices can be accessed through SSH or serial communication on the computer. Besides, A GUI is designed for displaying system status in real time. Debugging and testing tools are used such as Keil, DataCenter, and RF Explorer.

Chapter 3 No-Fault Benchmarking

Four single-interface tests are presented in this chapter to demonstrate a basic performance of testbed under normal conditions. Only probe frames are used in benchmarking tests. Tests are performed separately and independently on Ethernet, Wi-Fi, CAN, and BLE. In Ethernet and Wi-Fi benchmarking tests, ECU_a and ECU_b are active in the backbone network and only 28-byte probe frames are transmitted. In CAN and BLE benchmarking tests, ECU_a and DCU_{a1} are active in the device subnet and the sizes of probe frames are 108 bits for CAN and 144 bits for BLE.

3.1 Ethernet

ECU_a and ECU_b are active to transmit 28-byte probe frames via Ethernet switch in the backbone network. The transmission period of probe frames is 10 ms. The test running time is 20 seconds and around 2000 probe frames are transmitted in total. RTL, PDR and jitter are measured and all of them are calculated and stored on ECU_a. The result is shown in the Figure 4. For probe frame transmission, the average RTL is 287 μs without any packet drops. The standard deviation of RTL is 28 μs, which is included in the Table 2 in the following section summary. Periodic bulges on the RTL and the standard deviation curves can be observed. This periodic rise shows up in most of the following interface benchmarking tests. One of the reasons for this could be the OS task scheduling.

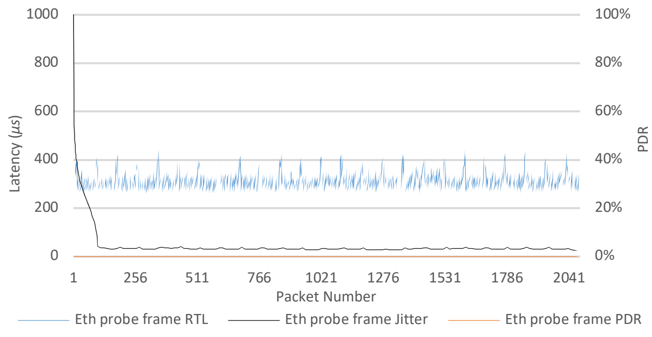


Figure 4. Ethernet Benchmarking Test

3.2 Wi-Fi

ECU_a and ECU_b are active to transmit 28-byte probe frames over customized Wi-Fi router. Tests on channels in both 2.4 GHz and 5 GHz are carried out, and results are shown in Figure 5 and 6, respectively. The channel bandwidth of both bands is 20 MHz. The transmission period of probe frames and running time is the same as the Ethernet test. Selected channels are Channel 11 in the 2.4 GHz band and Channel 44 in the 5 GHz band. Due to the limit of the testing environment, it is too hard to find a completely undisturbed channel in the 2.4 GHz band, but not in the 5 GHz band. Thus, the RTL and the corresponding standard deviation in Channel 11 has a pretty larger fluctuation than ones in Channel 44. The average RTLs of 2.4 GHz and 5 GHz are 1.717 *ms* and 1.552 *ms*, respectively, which are close to each other. However, the standard deviation of RTL in 2.4 GHz is 0.582 *ms*, which is larger than three times the one in 5 GHz of 0.169 *ms*. Besides, PDR in 5 GHz is around 0.05%, which is much lower than the PDR at 2.4 GHz of 0.3%. Moreover, there is a periodical rise of RTL in both 2.4 GHz and 5 GHz tests. It is similar to what the result of Ethernet benchmarking test reveals, which is considered as OS tasking delay.

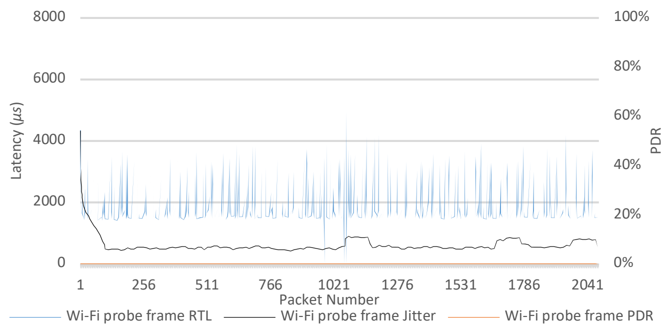


Figure 5. 2.4 GHz Wi-Fi Benchmarking Test

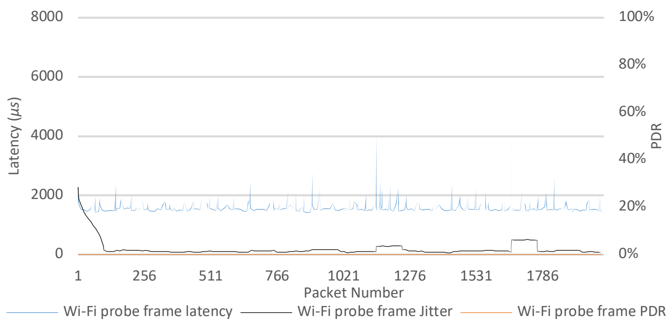


Figure 6. 5 GHz Wi-Fi Benchmarking Test

3.3 CAN-bus

ECU_a and DCU_{a1} are active to transmit 108-bit probe frame. The transmission period is 200 *ms*, and the bitrate of CAN-bus is set to 500 kbps. Probe frames are generated in ECU_a and transmitted to the DCU_{a1} to get echoing back over CAN. A CAN sniffer is used to monitor the traffic on CAN-bus, which is connected to the computer with USB. The result is shown in Figure 7. The average RTL of CAN probe frames is 751 μs, and its standard deviation is 13 μs, which means transmission over CAN-bus is very stable and reliable under no-fault condition. Similar to benchmarking tests of backbone networks, there is a periodical rise on the RTL within a range of 40 – 50 μs, which infers the OS tasking delay.

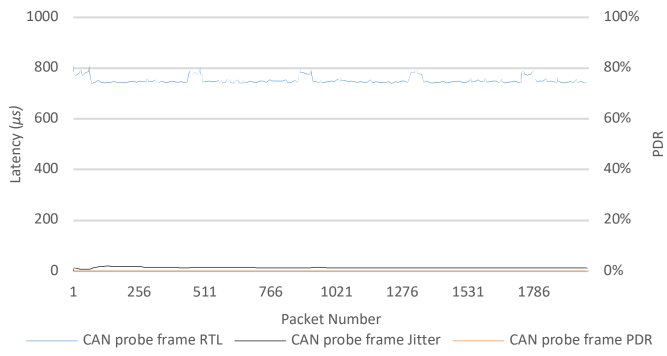


Figure 7. CAN-bus Benchmarking Test

3.4 BLE

Three tests are conducted to calculate the RTL of 100 packets of 8 bytes payload from the ECU to DCU with 0.5-meter separation and RSSI is also tested for each packet transmitted, as shown in Figure 8 and 9, respectively. The average RTL was equal to 110.6 *ms* with 27.9 *ms* jitter and -60.68 dBm average RSSI. The BLE RTL is mostly between 94 *ms* to 98 *ms*, but sometimes it goes to 142 *ms* to 146 *ms* or higher. The reason could be the retransmissions at a low level of the BLE stack.

Figure 8. BLE Benchmarking Test - Latency

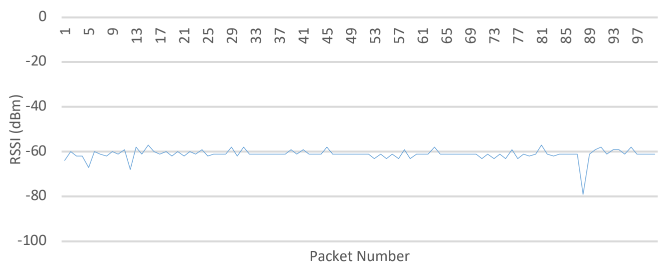


Figure 9. BLE Benchmarking Test - RSSI

3.5 Summary

A summary of testbed single-interface benchmarking tests is presented in Table 2. Besides PDR, average RTL and jitter which are mentioned above, a preliminary result of throughput with ACKs is carried out as well. Ethernet has a maximum throughput of 237.9 Mbps. In 5 GHz Wi-Fi, throughput tests are performed in HT20 and HT40 modes with channel bandwidths of 20 MHz and 40 MHz with IEEE 802.11n standard, and the results are 24.0 Mbps and 38.9 Mbps, respectively. In 2.4 GHz Wi-Fi, only HT20 mode is supported in IEEE 802.11g standard, and its throughput is 27.4 Mbps, which is a little higher than the one of HT20 in 5 GHz. The throughput of CAN is 131.9 kbps with 500 kbps bitrate. BLE uses error detection and retransmission in low layers which is not controllable in the program, so the drop rate at the API level appears as 0%, but it makes the average RTL is relatively low. BLE throughput tests are 1157 bps and 8712 bps with and without ACKs, respectively.

Protocol	PDR	$\overline{RTL} \pm stdev$ (ms)	Throughput (with ACKs)
Ethernet	0%	0.287 ± 0.028	237.9 Mbps
5 GHz Wi-Fi	0.05%	1.552 ± 0.169	24.0 Mbps (HT20) 38.9 Mbps (HT40)
2.4 GHz Wi-Fi	0.3%	1.717 ± 0.582	27.4 Mbps (HT20)
CAN	0.01%	0.751 ± 0.013	131.9 Kbps
BLE 4.1	0%	110.643 ± 27.979	1157 bps 8712 bps (no ACKs)

Table 2. Testbed Benchmarking Test Summary

Chapter 4 Algorithm Design

The hybrid network algorithm is designed to guarantee the stability and reliability of transmissions in the entire network system. It is transparent to applications. Network state estimator is the first step to implement the algorithm, as shown in Figure 10. In the network state estimation, probe frames are periodically sent on each interface which receiving devices are programmed to echo. RTL, PDR, and Jitter can be measured based on probe frame transmissions. Besides, RSSIs of Wi-Fi and BLE are also recorded. By experimenting in the normal case, threshold values of PDR, RTL and jitter can be determined, which are used to detect the current channel states. Depending on the channel state, traffic class reduction will be performed if the channel state degrades and channel switching will happen when it is becoming worse.

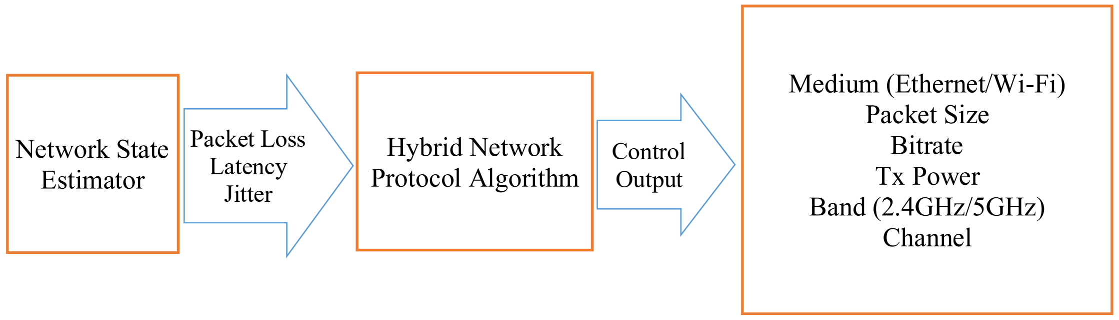


Figure 10. Hybrid Network Protocol Algorithm

4.1 Traffic Class Reduction

Traffic class reduction is performed in backbone networks. The basic idea is to ensure effective transmissions of high-priority data by sacrificing low-priority data when the channel state degrades. In the prototype, it assumes that DCU packets have three classes: A, B and C, and their priorities are from high to low, respectively. It simulates messages from different systems on vehicles, such as powertrain, driving assistance and infotainment. Rules of traffic class reduction are demonstrated in Table 3, which shows traffic classes in transmission under different channel states. In the normal case, ECU_a sent all data frames redundantly on both Ethernet and Wi-Fi. The reason to have redundancy is to enhance the reliability of transmission. Then, on the ECU_b of the receiving side, the redundancy will be filtered out by checking the sequence number and the rest of data will be sent to DCUs on the subnet. If channel state degrades, that is, when jitter exceeds the threshold for a certain time, class C will be dropped in most cases, but it will be carried on Wi-Fi and discarded on Ethernet if both Ethernet and Wi-Fi degrade at the first time. If the situation does not get better after dropping class C, which means that jitter is still higher than the threshold, class B will be aborted as well. If it gets worse or transmission failure happens, all data transmission will be terminated, and channel switching will be executed. Meanwhile, a preliminary recovery mechanism is designed for the situations that the current transmission channel is getting better. The recovery threshold is a little bit lower than the reduction threshold because it is expected to be more stable if it gets back to a better channel state. To prevent the threshold detection to be over sensitive, a buffer is set between the reduction threshold and the recovery threshold. If most parameters are located within the buffer, the channel state would keep the current state. Thus, the sensitivity of class-based reduction can is flexible to be modified in different situations and adjusted as needed.

Ethernet / Wi-Fi	Normal	Jitter	Extra Jitter	Ethernet Failure
Normal	E: A/B/C W: A/B/C	E: A/B W: A/B/C	E: A W: A/B/C	E: - W: A/B/C
Jitter	E: A/B/C W: A/B	E: A/B W: A/C	E: A W: A/B	E: - W: A/B
Extra Jitter	E: A/B/C W: A	E: A/B W: A	E: A W: A	E: - W: A
Wi-Fi Failure	E: A/B/C W: -	E: A/B W: -	E: A W: -	E: - W: -

Table 3. Class-based Traffic Reduction Algorithm

4.2 Channel Switching

The channel switching is divided into two parts: wired and wireless network switching and Wi-Fi channel switching. Wired and wireless network switching can be also called primary and secondary network switching. When the data transmission failure occurs on wired networks, it will switch to wireless networks. Channel switching between Ethernet and Wi-Fi is implemented by using redundant data transmission. Redundant data sent over Wi-Fi will be processed as currently received data when the data transmission over Ethernet fails. In the device subnet, data transmissions will be performed on BLE if CAN is disconnected. Wired and wireless network switching is a hot switching which can be processed when data transmission is on. Wi-Fi channel switching means that the Wi-Fi data transmission will hop to other available Wi-Fi bands and relative channels if the transmission in the current channel fails. The Wi-Fi channel switching process is shown in Figure 11. Wi-Fi status of ECUs is monitored by the Wi-Fi router. If congestion or interference is detected and the transmission issue persists after traffic class reduction, the Wi-Fi router will change the channel within the current band at first and ECUs will follow the change. If it doesn't get better, the router will switch to the other band. What needs to be noted is Wi-Fi protocols do not allow it to switch channels when there are any connections on so that the interface must be reinitialized and reconfigured if Wi-Fi channel switching is performed, which means that Wi-Fi channel switching is a cold switching. After the Wi-Fi interface reconfigured, ECUs will reconnect to each other by active detecting the same SSID of router automatically. In the prototype, since all processes of Wi-Fi channel switching is operated on the user level without programming in lower level protocols, it will take around 15 seconds. This long operation period makes it become the last choice of the hybrid network protocol algorithm.

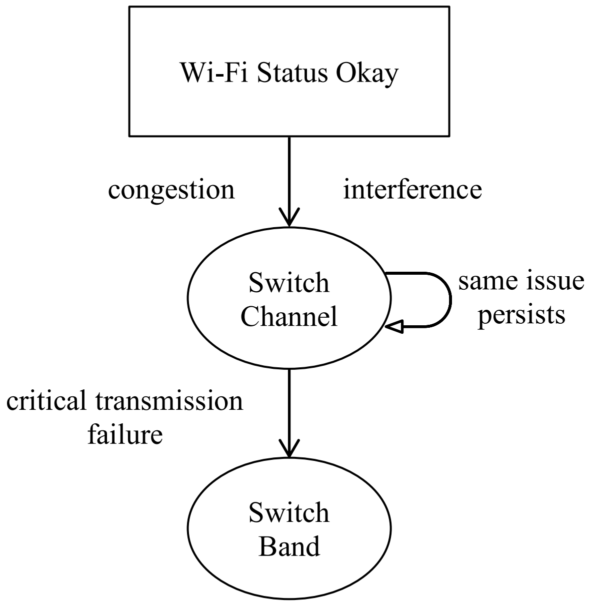


Figure 11. Wi-Fi Channel Switching

Chapter 5 Fault Detection

In this chapter, several experiment results are carried out under different conditions. By monitoring network state variables including RTL, PDR, and jitter, some network faults can be detected. In order to develop fault detection rules, the following faults are manually injected: Ethernet intermittent disconnection, Ethernet congestion, Wi-Fi congestion, Wi-Fi interference, CAN intermittent disconnection, and CAN congestion. Experiments on intermittent disconnections on wired networks are proposed to simulate the bad contact which is probably caused by the strong vibration or slight impact of the cables in vehicles. Wi-Fi congestions and interference tests aim to figure out how Wi-Fi transmission performs if the current channel is degraded or blocked by itself and another signal source. They are mainly about considerations of safety and reliability. The following subsections demonstrate experimental setups and relevant results.

5.1 Ethernet Intermittent Disconnection

Experimental Setup:

Two ECUs are connected to each other over an Ethernet switch using CAT7 Ethernet cables, and only the probe frames are transmitted over Ethernet with using UDP sockets. A static IP address is set for each ECU: 192.168.1.10 of ECU_a and 192.168.1.11 of ECU_b, which ensures that both ECUs are on the same subnets.

Test Procedure:

ECU_a is set to be as the server and ECU_b acts as the client. The server is for sending probe frames and the client is for echoing back what it received as soon as possible. The frequency of probe frame transmission is set at 100 Hz. The transmission program is running for 30 seconds. That is, around 3000 packets go through the experiment in total. While the program is running, the Ethernet cable will be manually disconnected and reconnected from the Ethernet interfaces on ECU_a or ECU_b as fast as possible.

Result Analysis:

Four Ethernet intermittent disconnections are intentionally generated during the experiment, which is recorded in places where the RTL equals zeros in Figure 12. The average RTL is 325 *μs,* excluding RTLs of dropped packets which are counted as zeros. PDR increases linearly at the same time. Since PDR is calculated with counting in the last 100 packets, the PDR will stay on a certain value for a while after Ethernet reconnected. The longer the disconnection time is, the greater the PDR stays. It can be obvious that normally the PDR is from around 10% to 27% when Ethernet intermittent disconnections appear. During Ethernet intermittent disconnection, ECUs still could hold their IP address for a short period after disconnection. It was observed in the Ethernet configuration on ECUs by using a Linux command 'ifconfig' during the test run. However, 100% of PDR happens as well and it takes more time to recover. A reasonable explanation is that if Ethernet disconnection exceeds a certain time, ECUs will lose the IP address. It needs time to be reassigned the IP address when ECUs reconnect to the Ethernet switch. In this case, PDR will go up to 100%, as shown in Figure 12 where the packet number is in the range of 794 to 1681 and 2101 to 2941. This can be considered as Ethernet failures. It indicates that when Ethernet failure happens, the period that reconnection will take is more than 8 *s*, which is a totally different case from the Ethernet intermittent disconnection. As a result, cases like around 20% or less PDR periodically happening on Ethernet can be considered as Ethernet intermittent disconnections. If PDR is larger than 30%, it can be considered as the Ethernet failure. PDR from 20% and 30% can be considered as a buffer zone where both situations could happen in the future.

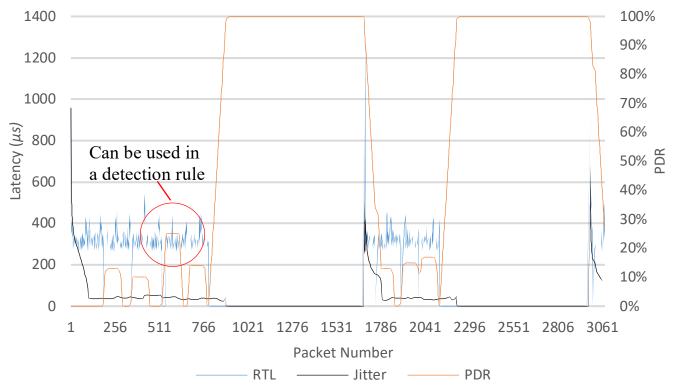


Figure 12. Ethernet Intermittent Disconnect Test

5.2 Ethernet Congestion

Experimental Setup:

Two ECUs are connected to each other over an Ethernet switch using CAT7 Ethernet cables. Probe frames and background traffic frames are used with UDP socket implemented. They are put in multiple threads separately. The size of probe frames is 28 bytes, and the size of the payload of background traffic frames varies from 0 bytes to 1400 bytes. The maximum transmission unit (MTU) is 1500 bytes which are the default setting on the Ethernet switch. IP addresses are the same as ones used in the Ethernet intermittent disconnection experiment.

Test Procedure:

ECU$_a$ and ECU$_b$ are set as the server and client, respectively. The server is for sending probe frames and the client is for echoing back what it received as soon as possible. Meanwhile, the server generates background traffic frames as well. The transmission frequency of the probe frame is set at 100 Hz, and background traffic frame is set at 1 kHz.

Result Analysis:

The result is shown in Table 4 and Figure 13, which illustrates that there is a positive correlation between the payload size of background traffic frames and the average RTL of probe frames with a certain transmission rate for both. Obviously, with the background traffic payload growing, average RTL of probe frames rises with no packet drops, and the standard deviation of RTL increases as well. Compared with the payload below 500 bytes, the average RTL has a larger increasing rate when the payload is over 500 bytes. However, faster growth on the standard deviation appears after 1000 bytes payload. Thus, the congestion can be noticed when the average RTL is larger than around 400 μs and the jitter is larger than around 35 μs. Due to there is no data frames are sending on the network, the situation could be a little different in actual data transmissions.

Payload	PDR	Round-trip Latency (μs)			
		Min.	Max.	Avg.	STDEV
0B	0%	263	580	287	28
100B	0%	260	556	313	26
500B	0%	261	861	322	30
1000B	0%	337	642	392	32
1400B	0%	368	1054	425	39

Table 4. Ethernet Congestion Test

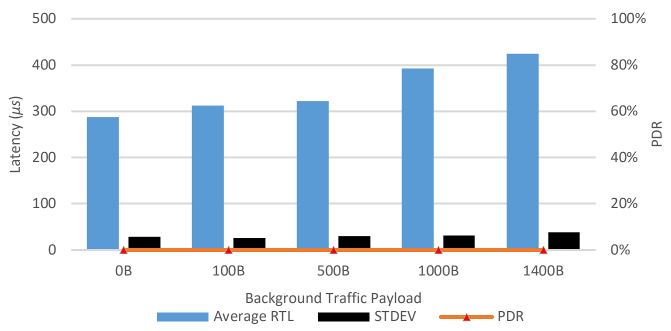


Figure 13. Ethernet Congestion Test

5.3 Wi-Fi Congestion

Experimental Setup:

ECU$_a$ and ECU$_b$ are interconnected in server-client mode over Wi-Fi using the customized Wi-Fi router. Their IP addresses are static as well, which are 192.168.2.10 and 192.168.2.11. In the Wi-Fi congestion experiment, a spectrum analyzer is used for scanning 2.4 GHz and 5 GHz spectra for clean channels and capturing bandwidth usage changes as congestion throughput increases. Two extra RPis are interconnected with each other in the ad-hoc mode as Wi-Fi congestion generators, which are used to generate background traffic frames with 1400-byte payload in the same channel that Wi-Fi router and ECUs are using. MTU is set as 1500 bytes by default on the Wi-Fi router, ECUs, and Wi-Fi congestion generators.

Test Procedure:

The test is divided into two scenarios: 2.4 GHz and 5 GHz. In the 2.4 GHz Wi-Fi congestion test, all devices are set in the same Wi-Fi channel in 2.4 GHz band, including Wi-Fi router, ECUs, and Wi-Fi congestion generators. The transmission frequency of the probe frame is 100 Hz, and the transmission frequency of the background traffic frame is set from 100 Hz up to 10 kHz. 5 GHz Wi-Fi congestion test has similar test procedure, but with all devices on the 5 GHz band. Besides, 2.4 GHz Wi-Fi uses HT20 mode and 5 GHz Wi-Fi uses both HT20 and HT40 mode in Wi-Fi experiments. The reason that HT40 mode is not adopted in 2.4 GHz is very limited channels in 2.4 GHz which means there is only one non-overlapping channel. It is hard to get an idle channel of 2.4 GHz band in HT40 mode for the experiment because many channels are used by university wireless access applications and other laboratories all the time.

Result Analysis:

The experiment results analysis on 2.4 GHz and 5 GHz are discussed separately below.

5.3.1 2.4 GHz Wi-Fi Congestion Test

Channel 3 (2.422 GHz) are used in 2.4 GHz Wi-Fi congestion test. The spectrums under different throughputs of background traffic frames are shown in Figure 14. The usage of the bandwidth increases as throughput increases. Derived from these spectrums, bandwidth usage has a higher increasing rate under 11.2 Mbps but a lower increasing rate above 11.2 Mbps. One of the reasons for this phenomenon is because it is reaching the bandwidth limit. Channel status is shown in Table 5 and the corresponding histogram is shown in Figure 15. As congestion throughput growth, the average RTL and the standard deviation increases on probe frames. PDR remains around 3% until the congestion throughput reaches 22.4 Mbps where the PDR is 13.41%. Besides, the standard deviation on 22.4 Mbps is much higher than others, which significantly degrades the Wi-Fi transmission. Thus, for 2.4 GHz Wi-Fi, a preliminary decision rule can be determined that it can be considered as heavy Wi-Fi congestion if PDR is larger than 10% or jitter is higher than 1.5 *ms*.

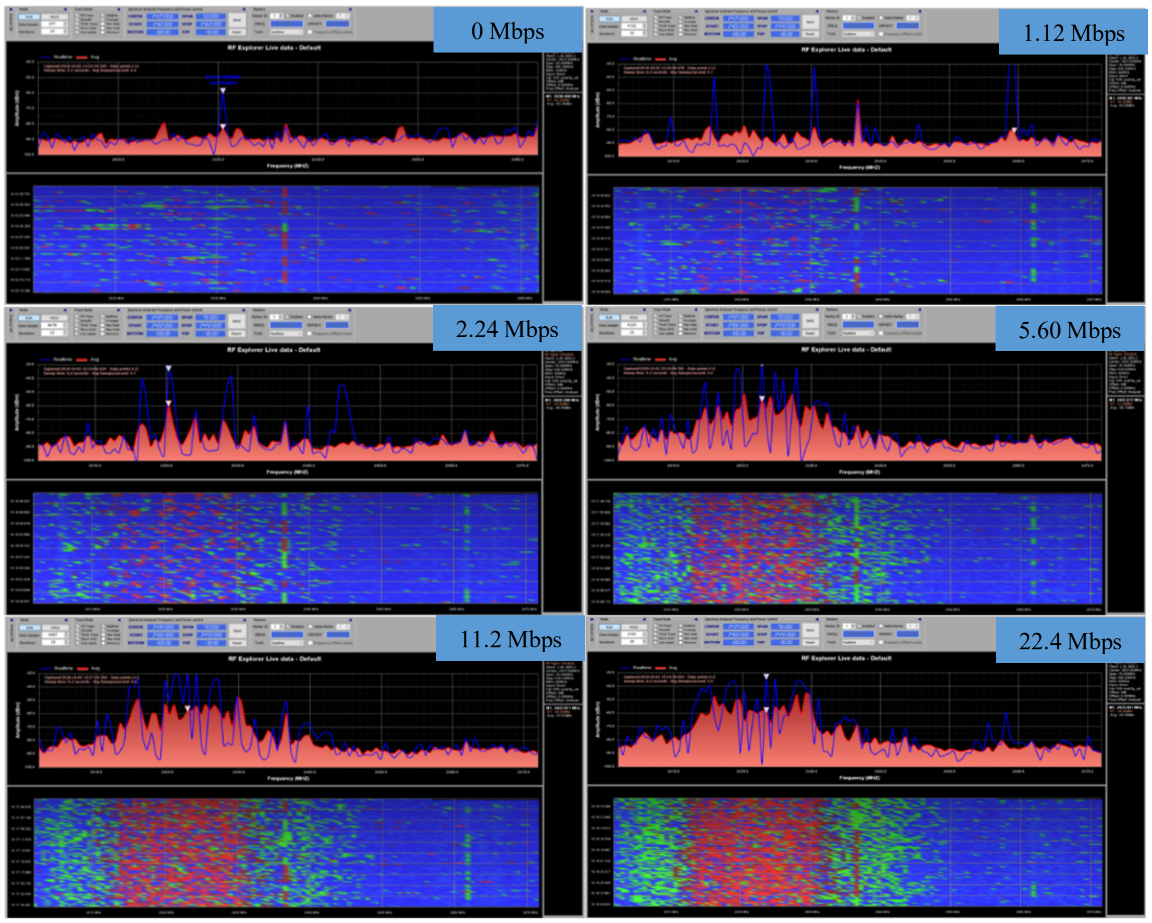


Figure 14. Spectra of 2.4 GHz Wi-Fi congestion tests with different background traffic data throughput. In each sub-picture, the upper half is the live spectrum, and the lower half is the waterfall plot of the spectrum.

Congestion Throughput	PDR	Round-trip Latency (*ms*)			
		Min.	Max.	Avg.	STDEV
1.12 Mbps	3.71%	1.336	7.871	1.903	0.961
2.24 Mbps	4.06%	1.222	7.992	2.361	1.191
5.56 Mbps	3.33%	1.343	7.933	2.606	1.561
11.2 Mbps	3.30%	1.347	7.972	2.986	1.298
22.4 Mbps	13.41%	1.409	7.986	3.918	1.993

Table 5. 2.4 GHz Wi-Fi Congestion Test

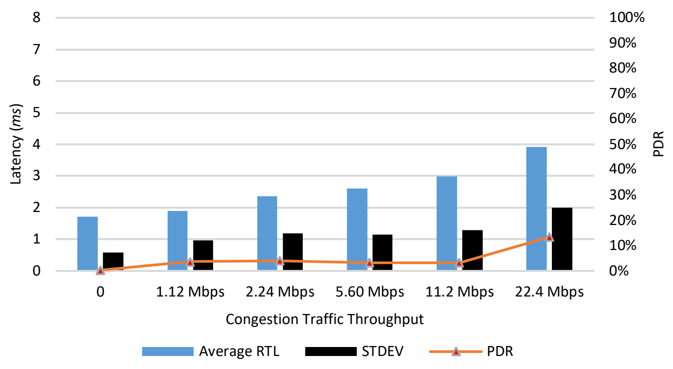


Figure 15. 2.4 GHz Wi-Fi Congestion Test

5.3.2 5 GHz Wi-Fi Congestion Test

Channel 44 (5.220 GHz) is used in 5 GHz Wi-Fi congestion test. Two experiments are conducted with two different Wi-Fi modes: HT20 and HT40. Wi-Fi channel bandwidth is 20 MHz in HT20 and 40 MHz in HT40. Two neighboring 20 MHz channels are bundled to form a 40 MHz channel. One channel is set as the main channel, and the other as the auxiliary channel. The main channel sends Beacon packets and data packets, and the auxiliary channel sends other packets. In Figure 16, the two pictures on the top are comparisons of the spectrum with the maximum throughput between HT20 and HT40 mode. The spectrum under different throughputs of background traffic frames is shown in the rest of Figure 16.

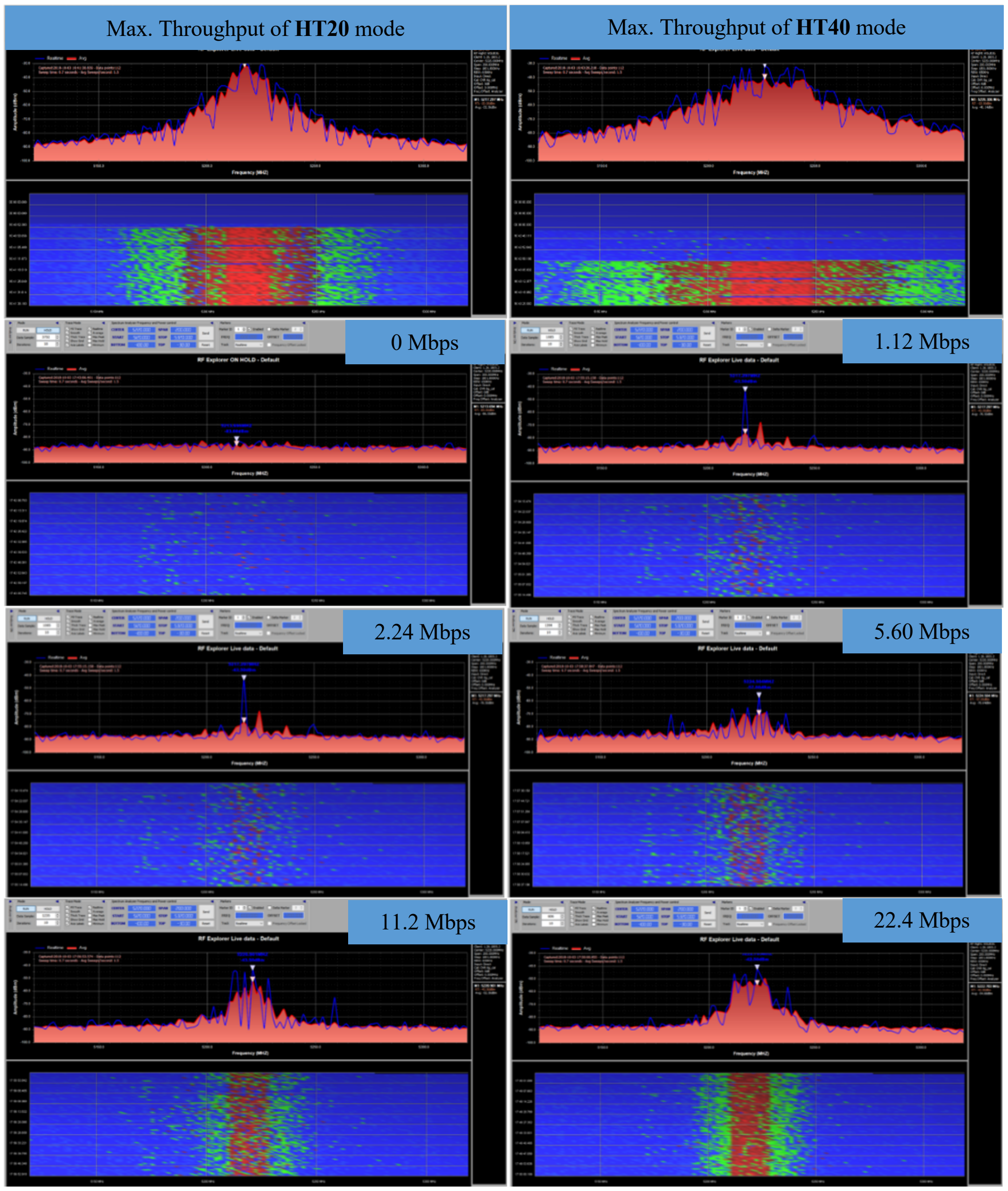


Figure 16. Spectra of 5 GHz Wi-Fi congestion tests with different data throughput

The congestion testing results of HT20 and HT40 are shown below. Performance of probe frames are presented in Table 6 and Table 7, and corresponding histograms are shown in Figure 17 and 18, respectively. In comparison between HT20 and HT40, HT40 slightly outperformed of HT20 below 11 Mbps, but it is much better than the HT20 above 11 Mbps. In HT20, significant increasing on PDR can be noticed when background traffic is above 11.2 Mbps. When congestion throughput is at 22.4 Mbps, 60% of packets are dropped in HT20, which is about 3 times of the PDR in HT40. The average RTL and the standard deviation of HT20 have a larger increasing rate than parameters of HT40 when more congestion traffic appears. This indicates that HT40 mode has more congestion tolerance than HT20 in 5 GHz Wi-Fi, especially when heavy congestion happens. It provides another way to avoid Wi-Fi congestion, but the premise is that the adjacent channel is clean and achievable. Figure 19 shows how congestion affects 5 GHz Wi-Fi with HT20 mode in the time domain. The test takes around 30 s, and the background traffic is injected at around 15 s with 4 Mbps throughput. There is an obvious difference in both RTL and jitter before and after 15 s. More fluctuations appeared on both after background traffic injected.

Congestion Throughput	PDR	Round-trip Latency (*ms*)			
		Min.	Max.	Avg.	STDEV
1.12 Mbps	0.30%	1.419	6.317	1.727	0.422
2.24 Mbps	0.50%	1.376	7.685	1.978	0.661
5.56 Mbps	0.70%	1.422	7.954	2.577	0.801
11.2 Mbps	28.90%	1.457	7.995	4.414	2.396
22.4 Mbps	60.00%	1.433	7.992	5.162	2.727

Table 6. 5 GHz **HT20** Wi-Fi Congestion Test

Congestion Throughput	PDR	Round-trip Latency (*ms*)			
		Min.	Max.	Avg.	STDEV
1.12 Mbps	0.06%	1.293	6.268	1.813	0.373
2.24 Mbps	0.26%	1.300	7.506	1.735	0.433
5.56 Mbps	0.61%	1.306	7.984	2.111	0.569
11.2 Mbps	7.66%	1.290	7.974	3.023	1.298
22.4 Mbps	19.65%	1.370	7.999	3.862	1.993

Table 7. 5 GHz **HT40** Wi-Fi Congestion Test

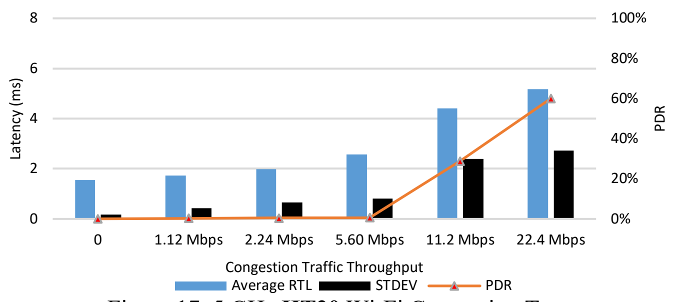


Figure 17. 5 GHz **HT20** Wi-Fi Congestion Test

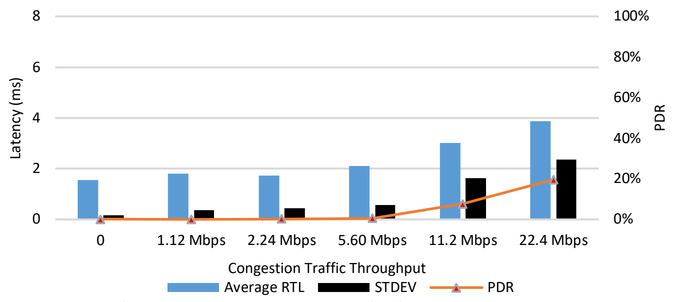


Figure 18. 5 GHz **HT40** Wi-Fi Congestion Test

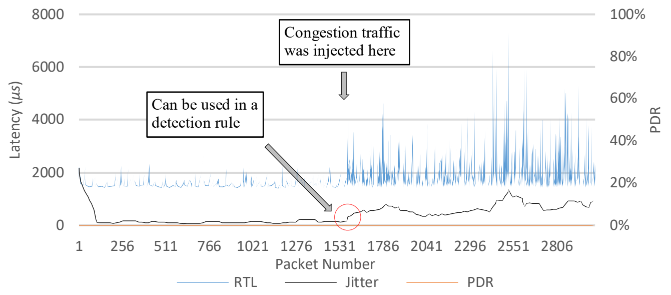


Figure 19. 5GHz HT20 Wi-Fi Congestion Test in Time Domain (Congestion traffic throughput = 4 Mbps)

5.4 Wi-Fi Interference

Experimental Setup:

A spectrum analyzer from RF Explorer is used to scan 2.4 GHz and 5 GHz spectra for clean channels and to capture bandwidth usage changes as background traffic throughput increases. Besides, an RF signal generator which is from RF Explorer as well, to create an interference signal which is in the same band and the same channel of the Wi-Fi router. Only probe frames are used for measuring and recording channel states. ECUs have the same setup as Wi-Fi congestion tests.

Test Procedure:

All devices are set in the same Wi-Fi channel, including Wi-Fi router, ECUs, and the RF signal generator. The transmission frequency of probe frames is set to 100 Hz. Tests are processed on 2.4 GHz and 5 GHz separately.

Result Analysis:

The interference testing result is analyzed based on two different experimental phenomena: transmission failure and no transmission failure. The transmission failure happens sometimes caused by interference signals and it is unpredictable. The ECUs lost their connections to the Wi-Fi router and their Wi-Fi IP addresses disappear as well when transmission failure happens. Unlike the spectrums of background traffic in congestion tests which spread all over the bandwidth, the interference signals are constrained at the central frequency. Theoretically, it mainly has an impact on the Wi-Fi carriers which are at the central frequency. During the experiments on both 2.4 GHz and 5 GHz, sometimes the interference can make the Wi-Fi transmission completely fails by moving relative position of the RF signal generator around the two ECUs, but sometimes it does not. The position of the RF signal generator which may cause the Wi-Fi failure is hard to be predicted due to limited resources. The testing results of 2.4 GHz and 5 GHz are carried out separately below.

5.4.1 2.4 GHz Wi-Fi Interference Test

Channel 3 (2.422 GHz) is used in 2.4 GHz Wi-Fi interference test. The spectra under different power levels of the RF signal generator are shown in Figure 20. The result of a 2.4 GHz Wi-Fi interference test without transmission failure is shown in Table 8. It indicates that there is no much difference from the no-fault condition within the average RTL when the power of interference signal increases from -30.4 dBm to 0.6 dBm. PDR is around 3% to 4%. There is no obvious growth law in both RTL and PDR, which means the interference hits a few of data transmission slightly and randomly. However, sometimes it could cause the transmission failure in a sudden when Wi-Fi interference signal is put close to ECUs with a high-power level of 0.6 dBm. It also makes ECUs lose their IP addresses. After turning off interference, ECUs would reconnect to the router and the transmission would get recovered as well.

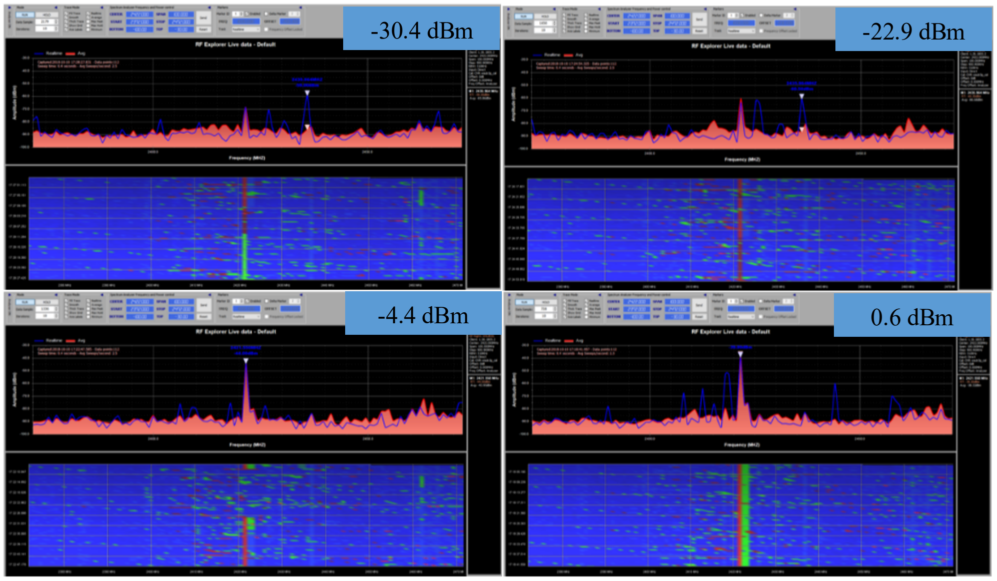


Figure 20. Spectra of 2.4 GHz Wi-Fi interference tests

Interference Signal	PDR	Round-trip Latency (*ms*)			
		Min.	Max.	Avg.	STDEV
-30.4 dBm	3.10%	0.981	7.956	1.586	0.943
-22.9 dBm	3.06%	1.079	7.971	1.569	0.913
-4.4 dBm	3.90%	1.087	7.973	1.594	0.954
0.6 dBm	4.02%	1.100	7.915	1.649	0.939

Table 8. 2.4 GHz Wi-Fi Interference Test

5.4.2 5 GHz Wi-Fi Interference Test

Channel 44 (5.220 GHz) is used in the 5 GHz Wi-Fi interference test. The spectra under different power levels of the RF signal generator are shown in Figure 21. The result of 5 GHz Wi-Fi interference test without transmission failure is shown in Table 9. The PDR and the average RTL do not change much when the power of the interference signal increases, but the jitter is increasing slightly in general. One of the reasons could be that the interference signal is spreading out from the central frequency as the power level increases, which can be noticed in the spectrum of from -17.3 dBm to -10.8 dBm. Moreover, the transmission failure could happen when an interference signal is put close to ECUs with high power levels like -13.8 dBm and -10.8 dBm, which is similar to the phenomena noticed in 2.4 GHz.

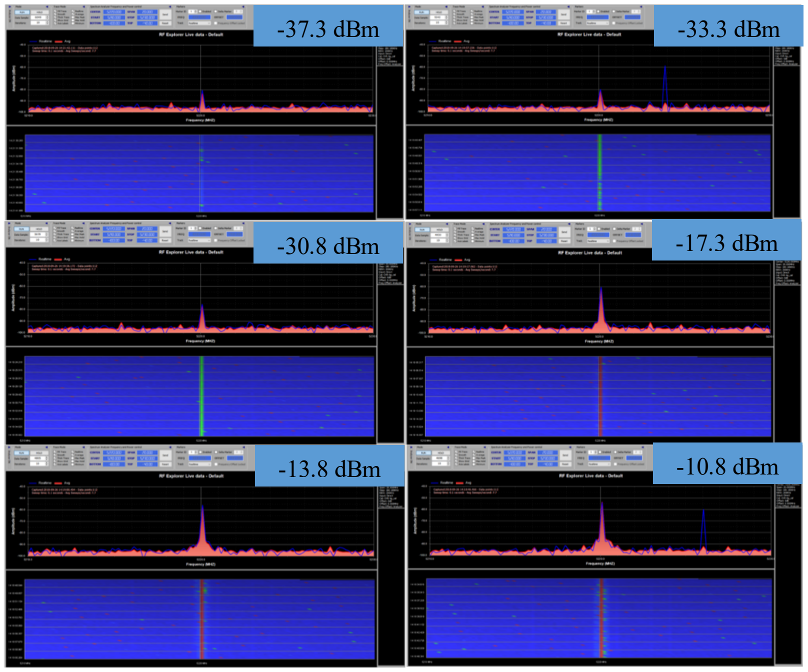


Figure 21. Spectra of 5 GHz Wi-Fi interference tests

Interference Signal	PDR	Round-trip Latency (*ms*)			
		Min.	Max.	Avg.	STDEV
-37.3 dBm	0.13%	1.350	6.419	1.613	0.283
-33.3 dBm	0.03%	1.436	4.806	1.619	0.237
-30.8 dBm	0.03%	1.215	7.223	1.622	0.330
-17.3 dBm	0.01%	1.326	6.951	1.612	0.414
-13.8 dBm	0%	1.340	7.159	1.607	0.305
-10.8 dBm	0.13%	1.352	7.981	1.652	0.391

Table 9. 5 GHz Wi-Fi Interference Test

5.5 CAN Intermittent Disconnect

Experimental Setup:

ECU$_a$ and DCU$_{a1}$ are connected to the same CAN bus. The bitrate of CAN-bus is set to 500 kbps. Only 108-bit probe frames are transmitted during the test running. SocketCAN and its library are used to support CAN-bus communications.

Test Procedure:

ECU$_a$ is the producer and DCU$_{a1}$ is the consumer. CAN transmission program sends around 5000 packets with a frequency of 100 Hz. While program running, disconnect and reconnect the CAN cable from the DCU as fast as possible.

Result Analysis:

The result of CAN intermittent disconnect test is shown in Figure 22. The average RTL excluding the dropped packets is 752.5 μs. Normally, the jitter is constant within 10 to 30 µs, and no packet drops. When intermittent disconnection happens, the jitter almost increases to 400 μs, and PDR goes up to 100% immediately. After reconnection, the jitter bounces back to the normal, as well as the PDR, which takes about 1.1 *ms* by estimation. It can be used as a detection rule to monitor if CAN intermittent disconnection happens.

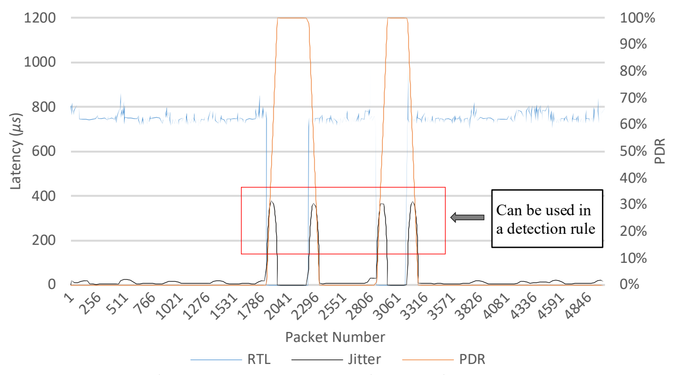


Figure 22. CAN Intermittent Disconnect

5.6 CAN Congestion

Experimental Setup:

ECU$_a$ is connected to DCU$_{a1}$ and DCU$_{a2}$ with CAN-bus in the device subnet. ECU$_a$ and DCU$_{a1}$ measure CAN-bus state by transmitting probe frames periodically, and DCU$_{a2}$ is for generating background traffic. CAN-bus is set to 500 kbps.

Test Procedure:

DCU$_{a1}$ is set as a consumer, which is only listening to CAN messages generated by the ECU$_a$ and echoes them back as soon as possible. DCU$_{a2}$ is set as a producer, where it would be generating messages over CAN fill the bus using 4 different transmission rates of 500 Hz, 1000 Hz, 2000 Hz, and 4000 Hz, without any terminal printing for the same previous reason. The test is held twice, for all frequency ranges stated above, one where the ECU has the highest priority and one where the DCU producing the background traffic has the highest priority. This is achieved by changing the CAN ID. Each test runs for 2500 packets, while the DCU$_{a2}$ is initiated to inject traffic approximately halfway through. Data will be recorded on the ECU, and later examined.

Result Analysis:

Testing results are shown in the following figures. Testing results of the ECU with low priorities and high priorities are shown in Figures from 23 to 26 and from 27 to 28, respectively. In the low priority test, approximately 80 μs of increase on RTL happens after background traffic injected under 2000 Hz. 4000 Hz background traffic injection brings transmission failure where 100% PDR appears. That means 4000 Hz background traffic makes the CAN bus so busy that the probe frames which ECU send to DCU are in the starvation due to the low priority. The reason is that when background traffic frequency is up to 4000 Hz, which is 432 kbps for the size of the CAN frame is 108 bits, the usage of CAN-bus is up to 86.4%. It is much higher than the commonly required usage which is 30%. This issue does not exist in a high priority test. Background traffic has little impact on transmissions of probe frames with a high priority. Only RTL increasing appears after background traffic injected.

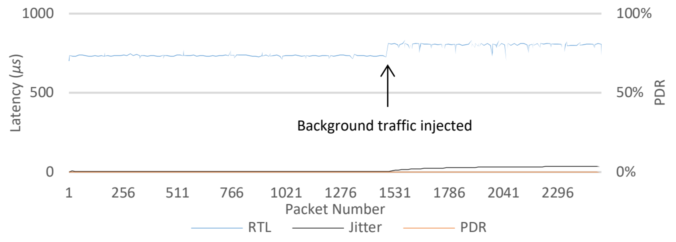


Figure 23. Low priority CAN message at 500 Hz

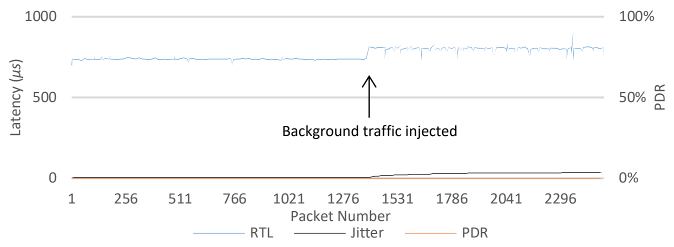


Figure 24. Low priority CAN message at 1000 Hz

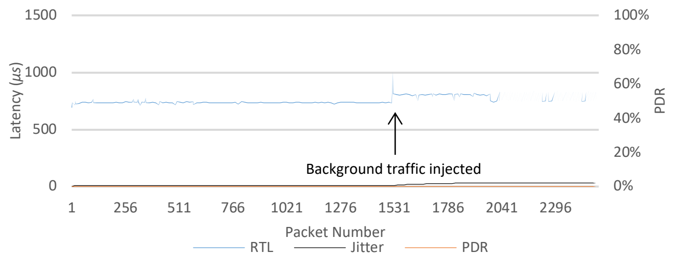


Figure 25. Low priority CAN message at 2000 Hz

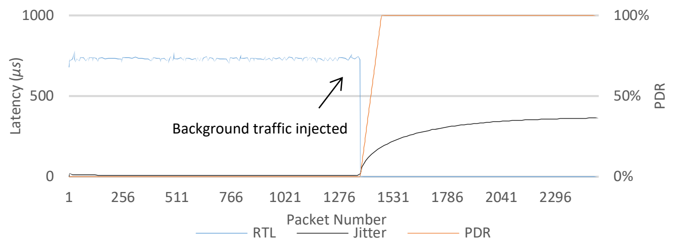


Figure 26. Low priority CAN message at 4000 Hz

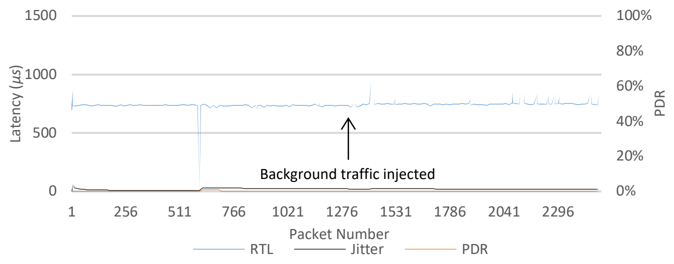


Figure 27. High priority CAN message at 2000 Hz

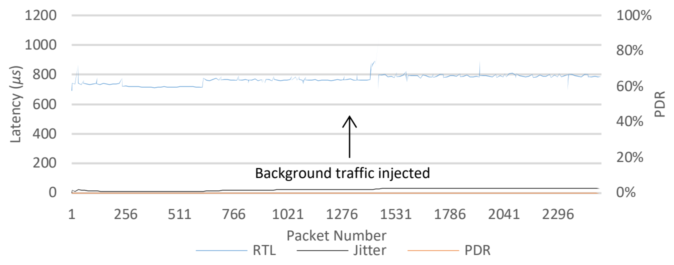
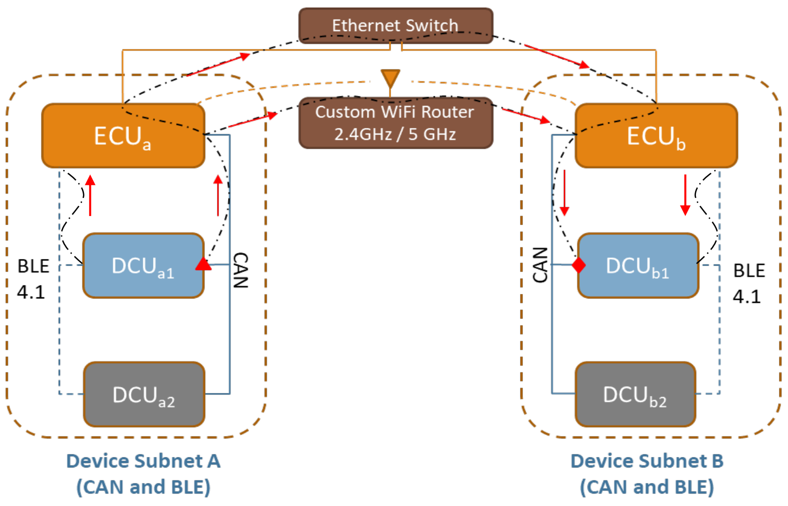


Figure 28. High priority CAN message at 4000 Hz

Chapter 6 Recovery Algorithm Experiments

In this section, the whole system is tested with a hybrid network algorithm enabled which is proposed in Section V. Experiments are implemented under the entire system running, which is set up as shown in Figure 29. The data frame transmission is following the red arrows. On the server side, DCU_{a1} is set as a producer to generate messages of three classes, and ECU_a is set as the server to receive the data from DCU_{a1} and forward them to ECU_b over both Ethernet and Wi-Fi. On the client side, ECU_b is set as a client to receive the data frames and forward them to the DCU_{b1} which is the end point of the data transmission. To enable the algorithm, thresholds are set on RTL, jitter, and PDR for both Ethernet and Wi-Fi but with different values, which is measured based on the previous experiments. If one of these parameters from the probe frames keeps larger than the threshold for 200 packets (2 seconds), the state will go to the next state, which means a worse channel status and traffic class reduction will be executed. The system would recover to the previous state if all parameters are stable under 2/3 (66%) of thresholds. Channel states from 1 to 4 are set for both Ethernet and Wi-Fi independently, and 1 standard for the normal condition and 4 is transmission failure. Channel states also can be mapped to Table 3 in Section V. The experiments of traffic class reduction and channel switching are separately introduced below.



Ethernet Switch
Custom WiFi Router 2.4GHz / 5 GHz
ECU$_a$
ECU$_b$
DCU$_{a1}$
DCU$_{a2}$
DCU$_{b1}$
DCU$_{b2}$
BLE 4.1
CAN
CAN
BLE 4.1
Device Subnet A (CAN and BLE)
Device Subnet B (CAN and BLE)

Figure 29. System Evaluation Setup

6.1 Class-based Traffic Reduction

Class-based traffic reduction is to guarantee high-priority packet transmission by sacrificing low-priority packets. It will occur when the channel state switches in the transmission. To induce channel states switching during the testing, ECU_a generates the background traffic frames with a ramped transmission rate which is used for injecting congestion into the backbone networks. The testing results of Ethernet and Wi-Fi are carried out separately in the following discussion.

6.1.1 Ethernet

In the Ethernet test, the transmission rate of ECU-generated background traffic is increased linearly from 10 kHz to 1 MHz during the experiment, and it is injected as well for all 3 message classes of 1000 bytes for each. The thresholds for state switching are set as 600 μs for RTL, 300 μs for jitter and 10% for PDR. The experiment is running for 120 seconds, and the background traffic is running for around 90 seconds, which is to be injected at 10 seconds after the test beginning.

The result is shown in Figure 30. The top is the chart of probe frame performance on Ethernet and bottom is the diagram of Ethernet states and the number of background traffic sent per second. The horizontal axis is the packet number of probe frames which is transmitted with 100 Hz. When background traffic is injected, the Ethernet state begins to switch between 1 and 2, which can be noticed when the 1100th of probe frames are transmitted. Transmissions on Ethernet start to be affected and degraded. As the ECU-generated background traffic increasing, the Ethernet jitter and latency becomes larger and packet drop appears, and state 3 and 4 shows up as well, which means the channel condition is getting worse. After background traffic finished, the Ethernet state goes back to the normal condition of state 1 immediately. The Ethernet state switching reflects the Ethernet channel condition in real time which the ECU relies on to make decisions on class-based traffic reduction. Because it ignores transmission of low-priority messages when Ethernet degrades, more bandwidth can be a server on the high-priority packets. Sometimes it makes the channel condition get better again and brings recovery on transmission. This is also the reason why the state bounced between adjacent values for many times after background traffic occurs.

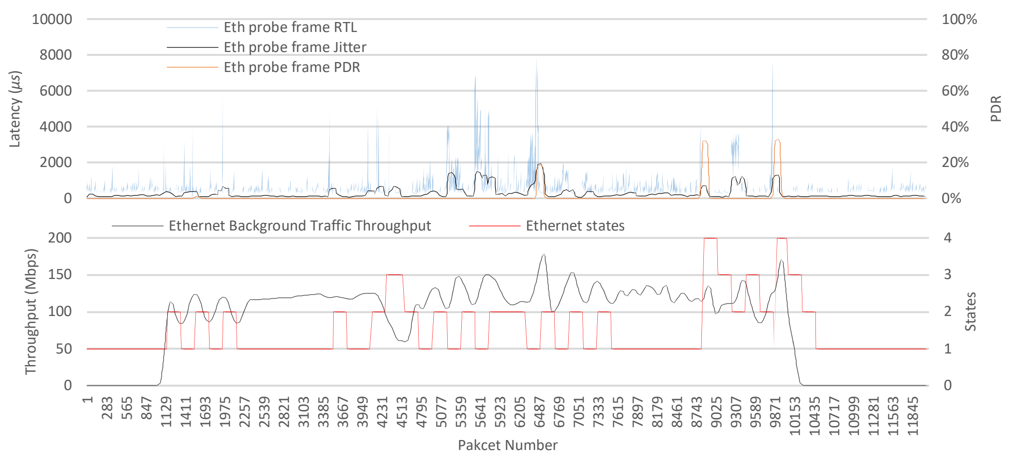


Figure 30. Class-based Traffic Reduction on Ethernet

6.1.2 Wi-Fi

The experimental setup for Wi-Fi is similar to the Ethernet but processed with different thresholds: 4000 μs for RTL, 1500 μs for jitter and 10% for PDR. Compared to the Ethernet, due to a relatively lower throughput of Wi-Fi, the transmission rates of ECU-generated background traffic are increasing linearly from 500 Hz to 50 kHz during the experiment. 5 GHz is used for the experiment because there is no available clean channel in 2.4 GHz.

The result is shown in Figure 31. The top is the probe frame performance on Wi-Fi and bottom is the diagram of Wi-Fi states and the number of background traffic sent per second. Same as Ethernet, Wi-Fi state starts to change after background traffic injected and bounces back and forth. However, compared to Ethernet states, Wi-Fi reaches state 3 earlier with a higher frequency, and Wi-Fi also sticks to state 4 for a long time with a high PDR when heavy congestion is raised up. It indicates that Wi-Fi has less congestion tolerance than the Ethernet.

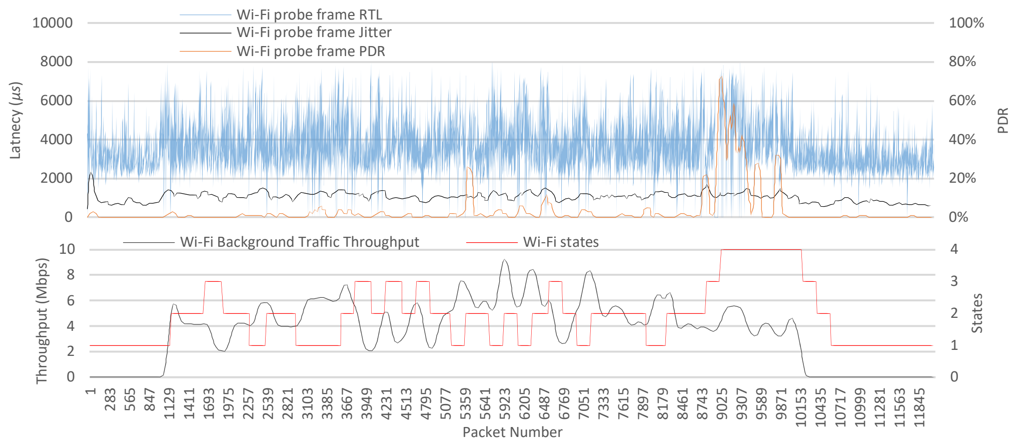


Figure 31. Class-based Traffic Reduction on Wi-Fi

6.2 Channel Switching

Experiments on channel switching are performed by using probe frames and DCU data frames without background traffic. Testing results of Ethernet and Wi-Fi switching, and Wi-Fi channel switching are introduced below.

6.2.1 Ethernet and Wi-Fi Switching

The first section of channel switching is Ethernet and Wi-Fi switching. It is transparent to the applications, which means applications do not need to define which interface is been used. It is implemented by using different subnet IP addresses, such as 192.168.1.x for the devices on the Ethernet and 192.168.2.x for the Wi-Fi. Since data transmission is guided by IP address, the interfaces can be simply notified. Due to the redundancy mechanism on backbone networks, the redundant data sent over Wi-Fi will come to the front when Ethernet failure happens. This forms the data transmission switching from Ethernet to Wi-Fi. In the experiment, data of three classes are generated by producer DCU with 10 Hz of generating frequency, so the number of DCU data frames that ECU_a forwards to ECU_b should be around 30 every second, theoretically. Testing results are shown in Figure 32 to 34. During the test, the Ethernet failure is artificially produced by unplugging Ethernet cables from ECUs. It can be pointed out in Figure 32 where the PDR of probe frames goes to 100% and in Figure 34 where the Ethernet state jumps to state 4 which is the worst state. The DCU data frames throughput goes down to 0 at the same time. Meanwhile, the Wi-Fi is working well, staying at state 1. The DCU data transmitting through Wi-Fi is not affected so that the client ECU can still receive the right number of DCU data frames. After reconnecting the Ethernet cables, the Ethernet state begins to step back to state 1, and DCU data transmissions on Ethernet is getting back to normal condition. It demonstrated that the hybrid network is Ethernet-fault tolerant in the prototype.

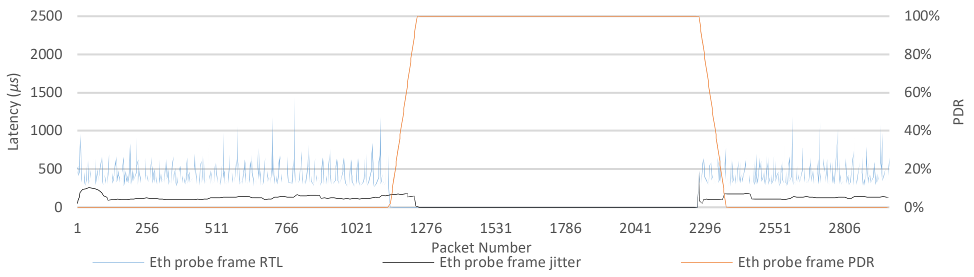


Figure 32. Ethernet and Wi-Fi Switching: Ethernet Probe Frame

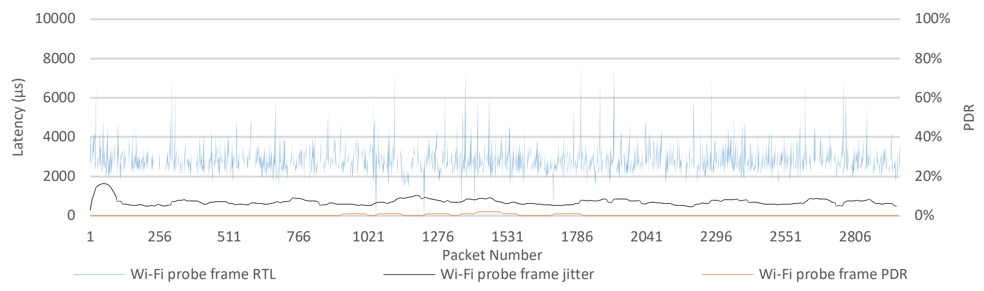


Figure 33. Ethernet and Wi-Fi Switching: Wi-Fi Probe Frame

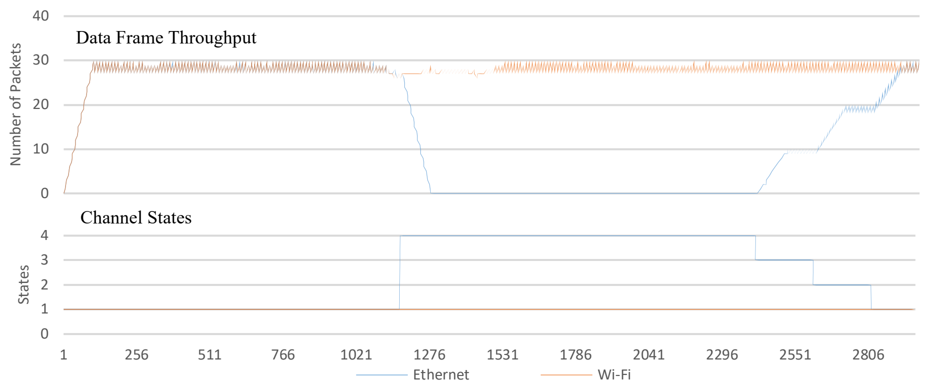


Figure 34. Ethernet and Wi-Fi Switching: Data Throughput and Channel States

6.2.2 Wi-Fi Channel Switching

The second part of channel switching is Wi-Fi channel switching. It has similar testing setup to Wi-Fi interference experiment. A program on customized Wi-Fi router is enabled for this testing to monitor the channel condition and switch Wi-Fi channels when transmission failure appears. Besides, this program will initialize the Wi-Fi to a certain band and channel at the beginning. Then it will scan all the Wi-Fi channels in the air to figure out which channel is clean and good to be reserved as a candidate for the future channel switching. It allows the router to choose a better channel dynamically. After finishing these processes, it will run into a while loop to keep checking the channel conditions in real time. Since the Wi-Fi channel switching only happened when a transmission failure occurs, a signal generator is used as interference to cut off Wi-Fi connections between two ECUs. There are two experiments: Wi-Fi channel switching within 5 GHz and Wi-Fi band switching, and the results are shown in Figure 35 and 36, respectively. Different PDR thresholds of Wi-Fi transmission failure are set: 30% for Wi-Fi channel switching within 5 GHz and 70% for Wi-Fi band switching. The PDR threshold of band switching is much higher because there are too many noises in 2.4 GHz and BLE is addressing at the beginning of the test, which may result in poor Wi-Fi channel state in 2.4 GHz band.

In channel switching within 5 GHz, the router initializes Wi-Fi to the channel 44 and two ECUs follow the Wi-Fi initialization. After Wi-Fi configured, probe frames transmission will start between two ECUs. Next, the signal generator is placed near the client ECU (ECU b) and set to the same channel. With the interference signal power increasing gradually, the PDR of probe frame transmission reaches the threshold of 30%, which triggers channel switching on the Wi-Fi router. The router chooses channel 36 as the new Wi-Fi channel and reconfigures Wi-Fi interface. During this period, ECUs loss their connections and their IP address as well because the router needs to reconfigure Wi-Fi interface, which can be marked where the PDR keeps 100% and RTL and jitter are both 0 in Figure 35. It takes around 20 seconds before ECUs reconnecting to the router. Once ECUs are reconnected to each other, the transmission will be continued, and parameters will go back to the normal condition. While the interference signal is still on, it does not affect the Wi-Fi transmission anymore. It can be informed by comparison before and after the channel switching by observing the jitter and PDR changes. There is no much difference in RTL during the whole test.

In Band switching, the router initialized Wi-Fi to channel 6 in 2.4 GHz. Similar to the channel switching within 5 GHz mentioned above, the test begins with probe frame transmission enabled. With the interference signal power increasing, the jitter and PDR in 2.4 GHz become pretty higher and reaches the PDR threshold to have band switching. Compared before and after the channel switching in Figure 35, jitter and PDR in 5 GHz is much better than ones in 2.4 GHz, and the RTL is improved as well after the band switched.

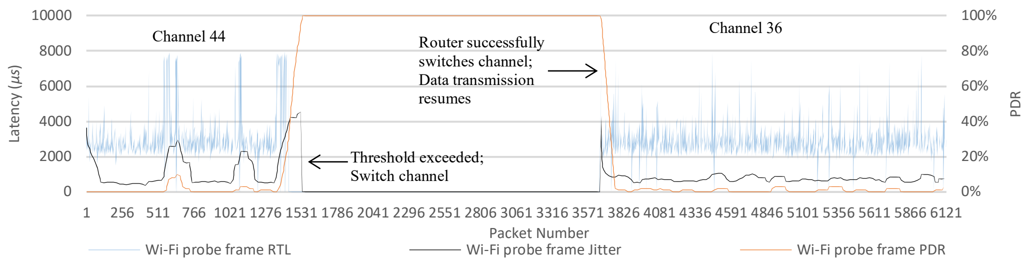


Figure 35. Wi-Fi Channel Switching within 5 GHz

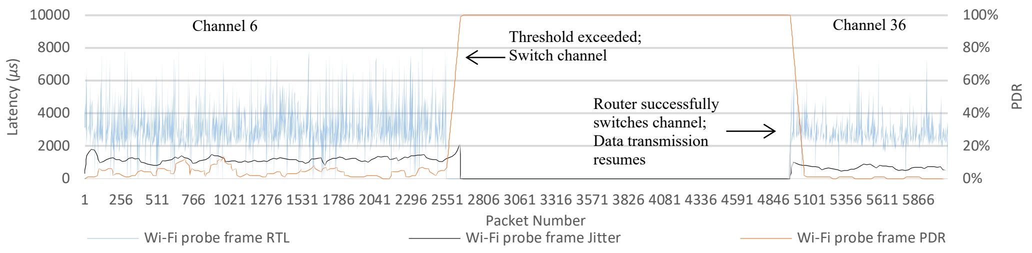


Figure 36. Wi-Fi Band Switching

Chapter 7 Summary

A new hybrid network prototype is designed and evaluated through experiments on the real testbed in this thesis. It uses a two-star topology with Ethernet, dual-band Wi-Fi, CAN and BLE. With this prototype, single-interface benchmarking tests and fault detection rules are processed. In addition, a hybrid network algorithm is developed based on a redundancy mechanism, which is the biggest achievement in this work. This algorithm is also tested using the testbed. With specific thresholds applied on latency, jitter and packet drop rate, the hybrid network system can perform the uninterrupted transmission by using the wireless network as backups when the wired network is disconnected or degraded. Moreover, a preliminary cold-switching algorithm is delivered on Wi-Fi channel switching to improve the Wi-Fi transmissions.

Under the no-fault condition, the end-to-end latency (DCU-ECU-DCU) is around 1 ms over Ethernet with CAN, 2.5 ms over 5 GHz Wi-Fi with CAN and 3 ms over 2.4 GHz Wi-Fi with CAN. However, Wi-Fi performance can be easily affected by channel conditions, which is not as stable as Ethernet. Besides, a hybrid network algorithm is proposed and implemented by fault detection experiments and congestion experiments to evaluate system performance. It includes traffic reduction based on class and Wi-Fi channel switching when needed. Both are demonstrated to be effective and improving network quality. Based on the system evaluation, Ethernet has more congestion tolerance than Wi-Fi. For Wi-Fi itself, compared to 2.4 GHz, 5 GHz has much better performance. However, heavy congestion would cause transmission failure (software-defined) on both Ethernet and Wi-Fi. Wi-Fi channel switching has a long period which is not friendly to the real-time system. BLE does not serve as a proper backup for CAN, that is at least when using a Raspberry Pi, where no to limited APIs are provided while the system developed provides relatively low throughput speed in comparison with the theoretical throughput speed.

In future work, a more sophisticated algorithm could be developed to optimize the hybrid network. Based on the algorithm proposed in this paper, different scenarios can be tested with realistic traffic running, and experiment with class-based traffic to find breaking points, the effectiveness of mitigation algorithms at maintaining network quality. After fault mitigation, recovery algorithms will be carried out. It will gradually reintroduce traffic while monitoring network state parameters. By using ramp congestion, it allows the system to estimate bandwidth remaining while network quality remains at acceptable levels. Dynamic thresholds will be also implemented to satisfy different situations or requirements. Besides, the research on Wi-Fi channel switching will go further to figure out if there is a better Wi-Fi adapter or applications which can shorten the switching time. BLE will be improved to carry the operation of the backup system in case of total failure routing messages throughout the whole network. Moreover, a third ECU can be joined into the system to develop a flexible routing algorithm and deploy time synchronization mechanism in WSNs.

References

[1] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi and L. Kilmartin, "Intra-Vehicle Networks: A Review," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, no. 2, pp. 534-545, April 2015.

[2] M. Faezipour, M. Nourani, A. Saeed and S. Addepalli, "Progress and Challenges in Intelligent Vehicle Area Networks," *ACM,* vol. 55, no. 2, pp. 90-100, Feb 2012.

[3] C. Carrizo and D. Dujovne, "Enhancing deterministic in-vehicle networks with a traffic management module," in *Embedded Systems (CASE), 2016 Seventh Argentine Conference on*, 2016.

[4] M. Cooperation, "MOST Cooperation," [Online]. Available: https://www.mostcooperation.com/technology/introduction/.

[5] H. Syed B. and J. Goldie, "An overview of LVDS technology," *National Semiconductor Application Note,* vol. 971, pp. 1-6, 1998.

[6] W. contributors, "IEEE 1394," Wikipedia, The Free Encyclopedia., [Online]. Available: https://en.wikipedia.org/w/index.php?title=IEEE_1394&oldid=886764026.

[7] W. Xing, H. Chen and H. Ding, "The application of controller area network on vehicle," in *Proceedings of the IEEE International Vehicle Electronics Conference (IVEC'99) (Cat. No.99EX257)*, 1999, pp. 455-458.

[8] M. Rahmani, R. Steffen, K. Tappayuthpijarn, E. Steinbach and G. Giordano, "Performance analysis of different network topologies for in-vehicle audio and video communication," in *2008 4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks*, Venice, 2008, pp. 179-184.

[9] J. Farkas, D. Fadyk, N. Finn, E. Gary, M. D. J. Teener, G. Parsons, P. Saltsidis and P. Thaler, "IEEE 802.1Q - Internet Engineering Task Force (IETF)," [Online]. Available: http://www6.ietf.org/meeting/86/tutorials/86-IEEE-8021-Thaler.pdf.

[10] W. contributors, "Audio Video Bridging," Wikipedia, The Free Encyclopedia., [Online]. Available: https://en.wikipedia.org/w/index.php?title=Audio_Video_Bridging&oldid=887056714.

[11] "AS6802: Time-Triggered Ethernet," SAE International.

[12] T. Steinbach, H. Lim, F. Korf, T. C. Schmidt, D. Herrscher and A. Wolisz, "Tomorrow's In-Car Interconnect? A Competitive Evaluation of IEEE 802.1 AVB and Time-Triggered Ethernet (AS6802)," in *2012 IEEE Vehicular Technology Conference (VTC Fall)*, Quebec City, 2012, pp. 1-5.

[13] H. Lim, D. Herrscher and F. Chaari, "Performance comparison of ieee 802.1 q and ieee 802.1 avb in an ethernet-based in-vehicle network," in *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, 2012.

[14] T. Steinbach, H. Lim, F. Korf, T. C. Schmidt, D. Herrscher and A. Wolisz, "Beware of the hidden! How cross-traffic affects quality assurances of competing real-time Ethernet standards for in-car communication," *2015 IEEE 40th Conference on Local Computer Networks (LCN),* pp. 1-9, 2015.

[15] T. Steinbach, F. Korf and T. C. Schmidt, "Comparing time-triggered Ethernet with FlexRay: An evaluation of competing approaches to real-time for in-vehicle networks," in *2010 IEEE International Workshop on Factory Communication Systems Proceedings*, Nancy, 2010, pp. 199-202.

[16] S. Tuohy, M. Glavin, E. Jones, C. Hughes and L. Kilmartin, "Hybrid testbed for simulating in-vehicle automotive networks," *Simulation Modelling Practice and Theory,* vol. 66, pp. 193--211, 2016.

[17] AUTOSAR, "AUTomotive Open System ARchitecture," 2007. [Online]. Available: http://www. autosar. org.

[18] K. Lakshmanan, G. Bhatia and R. Rajkumar, "Integrated end-to-end timing analysis of networked autosar-compliant systems," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2010.

[19] S. Anssi, S. Tucci-Piergiovanni, S. Kuntz, S. Gerard and F. Terrier, "Enabling scheduling analysis for AUTOSAR systems," in *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, 2011.

[20] N. Lu, N. Cheng, N. Zhang, X. Shen and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE internet of things journal,* vol. 1, no. 4, pp. 289--299, 2014.

[21] M. Ahmed, C. U. Saraydar, T. ElBatt, J. Yin, T. Talty and M. Ames, "Intra-vehicular wireless networks," in *Globecom Workshops, 2007 IEEE*, 2007.

[22] H. Yue, C. Zhang, P. Miao, Y. Fang and S. Chen, "A time-efficient information collection protocol for large-scale RFID systems," in *INFOCOM, 2012 Proceedings IEEE*, 2012.

[23] W. Niu, J. Li, S. Liu and T. Talty, "Intra-vehicle ultra-wideband communication testbed," in *Military Communications Conference*, 2007.

[24] J. Blumenstein, T. Mikulasek, T. Zemen, C. Mecklenbrauker, R. Marsalek and A. Prokes, "In-vehicle mm-wave channel model and measurement," in *Vehicular Technology Conference (VTC Fall)*, 2014.

[25] R. Frank, W. Bronzi, G. Castignani and T. Engel, "Bluetooth Low Energy: An alternative technology for VANET applications," in *2014 11th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, Obergurgl, 2014, pp. 104-107.