A study toward the practical use of WLAN-based vehicular network systems

Graduate School of Science and Technology,
Shizuoka University

Thesis

A study toward the practical use of WLAN-based vehicular network systems

5594–5006
Arata Kato

March 2022

Contents

List of figures

List of tables

Abstract

Vehicular networks realize cooperative awareness and collective perception and improve road safety, traffic management, and driving experiences. Under disaster conditions, vehicular networks can also be utilized as a communication system because they can work without standing communication infrastructures like cellular networks. Although the expectation for vehicular networks keeps growing, there are only a few precedents of deploying vehicular network systems to the real world due to expensive costs for installing network equipment to vehicles, implementating onboard units and their software, and legislating. Especially, operation validation of vehicular network systems cannot be easily performed in the real world due to high preparation cost of vehicles and network equipment and legal permission to use public roads.

There are two ways to validate the operation of the vehicular network system implementations: Field testing and network emulation. Field testing reveals the practical performance of the vehicular network system in the actual field. Therefore, the measurement results of the real field performance of vehicular network systems are invaluable for developing vehicular network systems, but there are not enough empirical reports, especially, the bulk data transmission performance in vehicular networks using new wireless LAN technologies. Although the bulk data transmission in the vehicular network is significant for sharing photos and videos in disaster conditions and disseminating in-car infotainment contents, it is difficult to estimate their practical performance and validate whether the evaluation results in a testbed match the practical performance.

Meanwhile, network emulation enables the vehicular network system to work in a virtual network in which a network simulator reproduces the vehicles' mobility and signal propagation. Almost all existing network emulators focus on link emulation based on Ethernet virtualization technologies such as a TUN/TAP device and allow a network

simulator to capture data packet flows using the TUN/TAP devices and apply the bandwidth limitation, delay insertion, and packet loss for the captured packet flow based on radio propagation and mobility models. However, the existing network emulators are insufficient to be used for vehicular network emulation because they cannot reproduce the behavior of the vehicular network protocols, such as IEEE 802.11p and ETSI Decentralized Congestion Control (DCC). The vehicular network protocols adaptively control transmission power based on received signal strengths to avoid interference, and their behavior is a significant factor for operation validation of the vehicular network systems. Therefore, it is important to emulate the behavior of the vehicular network protocols as they perform in the real world.

This dissertation presents two contributions to encourage developing vehicular network systems. The first contribution is a novel wireless network emulator with wireless network tap devices. The wireless network emulator can reproduce the behavior of the Linux IEEE 802.11 protocol stacks and network applications in a virtual network as they perform in the practical IEEE 802.11 network. The wireless LAN emulator allows the Linux IEEE 802.11 protocol stacks and a network simulator to exchange IEEE 802.11 frame flows and IEEE 802.11 device configuration flows, such as transmission power change and received signal strength notification, by using the wireless network tap devices, which is a virtual IEEE 802.11 device developed by the author. This study shows that the wireless network emulator can reproduce the behavior of an actual WLAN-based network by using the real Linux IEEE 802.11 implementation and network applications. This study also proposes some techniques to improve the real-time property of network emulation with the proposed emulator. The effectiveness of the techniques is validated through experiments.

The second contribution is empirical measurement of field performance of link setup time and bulk data transmission in WLAN-based inter-vehicular communications. This study presents the first empirical report of the initial link setup time reduction for IEEE 802.11-based inter-vehicular communication using Fast Initial Link Setup (FILS) and Wi-Fi Protected Access with Protected Extensible Authentication Protocol (WPA-PEAP). The performance evaluation reveals that FILS enables the initial link setup to complete within about 130 ms between vehicles with a high relative speed of 80 km/h, ten times shorter than WPA2-PEAP authentication, and transfer a maximum of 75 MB while the vehicles are passing each other.

Chapter 1 Introduction

This chapter describes the background of this study and a brief summary of the contributions of this dissertation.

1.1 Growing expectation to vehicular networking

Vehicular networks have attracted attention from academia, industries, and governments, with increasing interest in intelligent transport systems (ITS) and smart cities. Figure 1.1 shows examples of vehicular network applications. For example, the vehicular networks are expected to improve road safety by realizing cooperative awareness, collective perception, etc. Cooperative awareness[1] makes vehicles broadcast beacons, including the vehicles' position and velocity, and allows the vehicles receiving the beacons to detect the sender vehicles if the senders are at non-line-of-sight (NLOS) positions from the receivers. European Telecommunication Standards Institute (ETSI) standardizes the specification of cooperative awareness in ETSI EN 302 637-2[1], and some researchers report cooperative awareness is effective to reduce accidents based on simulation results[2]. Collective perception[3] makes vehicles to send sensor information to the other vehicles and allows the receiver vehicles to detect vehicles in their blind spots using the sensor information. The sensor information includes the photos captured by cameras, point group data from LiDAR (Light Detection and Ranging) devices, GPS (Glocal Positioning System) locations of neighboring vehicles, etc. If cooperative awareness and collective perception are realized, the number of traffic accidents can be reduced.

In a vehicular network, a connected vehicle, which is a vehicle equipped with a communication device, communicates with other connected vehicles or communication entities such as road-side units and personal mobile devices wirelessly. A vehicular network consists of various communication systems based on the classification of the communication entities, for example, Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrian (V2P), Vehicle-to-Network (V2N) communication systems. The communication systems used in a vehicular network are collectively called Vehicle-to-Everything (V2X) communication. Current two popular radio communication technologies used in vehicular networks are standardized in IEEE 802.11p[15] and LTE-V2X[16], which are based on Wi-Fi and Cellular communication, respectively.

IEEE 802.11p was standardized in 2012. It is based on IEEE 802.11a including modifications for road safety application and devices with high mobility. The road safety application is, for example, that a vehicle broadcasts beacons including their current positions and allows other vehicles not detecting the vehicle to recognize it and encourage avoiding a collision. The first version of LTE-V2X was standardized in 2018 and defines direct communication mechanism between vehicles without a relay via a cellular base station. In addition, next-generation radio communication standards for vehicular networks, IEEE 802.11bd[17] and 5G NR V2X[18, 19], are under establishment to improve a large amount of data transmission performance.

Vehicular networks can be used for not only road safety but also various network services using various communication systems. For example, vehicular networks can be utilized as a communication infrastructure in urban areas and are expected to serve intelligent navigation and optimize road traffic[20, 21]. V2I/V2N communication enable road-side units (RSUs) connected to a traffic control center and sensors equipment installed along roads to gather sensor information, such as photos capturing road traffics and GPS positions of vehicles and enable the traffic control center to monitor road traffic and navigate vehicles to clear up traffic congestions[22]. In addition, the U.S. Department of Transportation (U.S. DOT) indicates in [23] that clearing up traffic congestion with vehicular networking makes it reduce fuel consumption and benefits the natural environment. Vehicular networks are also beneficial to provide infotainment services to passengers in cars[24, 25].

Table 1.1: Vehicular network applications

System	Type	Description
Cooperative awareness[4, 5]	V2V	Connected vehicles disseminate their location and velocity to other vehicles and avoid collisions with them.
Collective perception[6]	V2V	Connected vehicles collect sensor information such as camera images and LiDAR/LADAR data of other vehicles via V2V and detect vehicles in blind spots.
Platooning/ Cooperative adaptive cruise control (CACC) [7, 8, 9]	V2V	A connected vehicle receives messages including the accelaration and velocity of another vehicle running ahead and adjusts its velocity to keep the distance between the vehicles or make a space for a vehicle approching from another lane.
Infortainment content dissemination[10]	V2I	Road-side units disseminate movies and advertisements to neighboring vehicles and provide infotainment contents to the vehicles.
Driving Safety Support System (DSSS)[11, 12]	V2V, V2I, V2P	Road-side units with sensors around intersections detect approaching vehicles and disseminate approaching warnings to other vehicles and pedestrians to prevent accidents.
Urban monitoing (MobEyes)[13]	V2X	A sensor network constructed with connected vehicles collects sensor information from the vehicles and use the sensor data to monitor traffic congestion, road surface diagnosis, atomosphere pollution analysis, etc.
Disaster Vehicular DTN[14]	V2X	Connected vehicles form a delay/disruption tolerant network (DTN) in disaster sites, and the vehicles carry data to isolated/radio blind areas where lost communication links such as cellular/optical fiber links by natural disasters to keep connectivity with the areas.

For example, broadcasting local advertisement content like commercial videos from shops along a road where vehicles are moving has a good impact on the economic effect in the local market.

As of 2021, the growing expectation for vehicular networks triggers several countries to propose the concept of a smart city integrated with vehicular networks in their national programs of science and technology[1] and competitively develop vehicular network systems. A smart city means a modern city that provides advanced citizen services with information and communication technology (ICT) systems for growing the benefit of citizens and businesses. In the smart city, vehicular networks are expected to provide communication infrastructures in urban areas and collect sensors data such as photos captured by on-board cameras and location information that are used to analyze and manage transport systems[26].

Vehicular networking is recently becoming a significant technology to improve communication under disaster conditions (disaster communication). The government of Japan mentions in Society 5.0[26] that they aim to utilize vehicular networks under disaster conditions because the vehicular network works without standing communication infrastructures such as cellular networks and optical fiber networks and has the robustness to these infrastructure failures. Meanwhile, Asia Pacific Telecommunity (APT) recommends using a vehicular delay/disruption tolerant network (VDTN) as a means of communication under disaster conditions[30]. The VDTN is one type of vehicular network. In the VDTN, the connected vehicles carry application data between remote places with no direct communication links. Even if natural disasters break the standing communication infrastructures, the connected vehicles keep communications between the remote places. Although the various systems using vehicular networks are proposed, there are some problems on deploying vehicular network systems to the real world.

[1]As of 2021, Japan, the United States, Europe, and China have proposed their concepts of a smart city in Society 5.0[26], Smart City Challenge[27], Industry 5.0[28], and Made in China 2025[29], respectively.

1.2 Problems on deploying vehicular network systems in the real world

As of 2021, various organizations have been actively working for implementing vehicular network systems for years. However, there are not enough precedents of deploying vehicular network systems to the real world[31, 32, 33, 12] due to the following reasons.

- There are not enough development environments for the software and hardware of onboard units and road-side units.
- Initial cost for installing network equipment such as onboard units and road-side units is expensive.
- Legal revisions about radio regulation and road traffic act are required to deploy vehicular network systems in public.

On developing the vehicular network systems' components, two technical barriers make it difficult to develop vehicular network systems. The first barrier is not enough operation validation environments for vehicular network systems. The second barrier is not enough empirical reports of the practical performance of bulk data transmission in vehicular networks due to the difficulty of conducting field testing.

Validating the operation of vehicular network systems is essential before deploying the systems to the real world. Since vehicular network systems will be used for traffic accident prevention, it is not preferrable that the systems malfunction and do not prevent a traffic accident. For example, vehicles periodically broadcasting basic safety messages are required to check channel congestion from received signal strengths and adjust transmission power level or interval of the messages to avoid signal interference with other vehicles. If the system malfunctions and causes interference of basic safety messages, a vehicle may not identify another vehicle approaching from its blind spot and cannot avoid a collision. To validate the operation of the transmission power/interval adjustment function, it is necessary to execute the program implementing the adjustment function and reproduce the received signal strength changes following vehicles' movement.

Operation validation of vehicular network systems is also important to keep compatibility and functionality of the systems. Vehicular network systems will be widely used across various types of vehicles from various manufacturers, and developers of vehicular network systems need to implement their system to fulfill the requirements necessary to keep compatibility and functionality of vehicular network systems components. For example, the Society of Automotive Engineers (SAE) international standardizes the minimum requirements of onboard units for safety applications in SAE J2945/1[34]. SAE J2945/1 benefits keeping reliability and interoperability between onboard units implemented by different manufacturers and requires developers to check the operation of the hardware/-software components according to the requirements. For example, SAE J2945/1 defines the minimum requirements for the IEEE 802.11 MAC and PHY services necessary for V2V applications. Therefore, it is essential to have a test environment for the actual software implementation of onboard units and enlarge the knowledge about the field performance of vehicular network systems.

There are two ways that realize operation validation of vehicular network systems using real implementations: field testing and network emulation. Field testing is the most effective evaluation method to reveal the functional performance of vehicular network systems. The performance of a vehicular network system is influenced by various factors such as vehicles' mobility, radio propagation, signal fading, line of sight between connected vehicles, and network topology, road conditions, etc. Field testing can reveal the field performance of vehicular networks including the influence of these factors. However, field testing is the most difficult method to perform due to some barriers such as initial monetary cost for vehicles and network equipment and legal permission to use the prototype system in public roads.

On the other hand, network emulation can be performed easier than field testing. Network emulation enables network applications in a virtual network in which a network simulator reproduces radio propagation and mobility. The network simulator typically utilizes virtual Ethernet technologies such as TUN/TAP devices to capture data packet flows from the network applications and applies bandwidth limitation, delay insertion, and packet loss for the captured packet flows. By changing the radio propagation and mobility models, the network simulator can reproduce the behavior of vehicular network systems. However, the existing network emulators depend on a specific network simulator and do not allow the network protocol stacks in the low-layers of an operating system, such as IEEE 802.11 MAC protocol implementation. According to SAE J2945/1, the IEEE 802.11 protocol stacks have an important role in the vehicular network systems to control signal congestions and avoid interference, and the operation validation of the IEEE 802.11 protocol stacks is significant for developing the vehicular network systems. As described above, it is important to validate the software operation of an onboard unit to provide users advanced traffic accident prevention services by vehicular network systems. The vehicular network emulation environment should be able to reproduce the behavior of onboard unit's software in the real environment with consideration of dynamics of vehicles' mobility. However, the existing network emulators comes from link emulation technology for wired networks such as Ethernet and are insufficient to emulate the behavior of the IEEE 802.11 protocol stacks.

It is also essential to understand of the field performance of vehicular network systems. Many researchers and developers have proposed various vehicular network systems and related technologies. The effectiveness of these proposed ones are often presented based on simulation results due to the difficulty of performing field testing. These simulation results should be compared with the actual performance measured in the real environment to check their validity because they can incompletely include the effect of some factors that are highly abstracted in simulation models, such as network equipment implementations and road environments. While emulation results have more reliable than simulation results because they partially come from some behavior of the real implementations, emulation results should also be checked with actual performance measured in the real environments. However, few reports of field performance of vehicular network systems make it difficult to identify whether these simulation/emulation results are close to the actual performance of vehicular network systems in the real environment.

For example, revealing the bulk data transmission performance in inter-vehicular communications is essential to transfer a large amount of data such as photos and videos between vehicles. If the inter-vehicular bulk data transmission performance is not identified, it is impossible to estimate how many photos and videos can be transfferred to use up the bandwitdh between the vehicles. On the other hand, if the bulk data transmission performance is identified, the sender vehicle selects data whose total size is the same as the bandwidth between the vehicles and effectively utilizes the bandwidth. In addition, the measurement results in the real fields are necessary to compare the network emulation results and check the validity of the network emulation results. However, field testing cannot be easily performed because it takes many preparation costs.

1.3 Contributions of this dissertation

This dissertation presents two contributions to encourage developing vehicular network systems. The first contribution is a novel wireless LAN emulator with wireless network tap devices (WiNE-Tap), which can be used for the operation validation system for network applications and IEEE 802.11 protocol implementation for Linux systems. As discussed above, the developers and researchers of vehicular network systems should be responsible for developing onboard units that meet the requirements for V2V applications. The implementation of the vehicular network systems can be validated by network emulation or field testing. Network emulation is beneficial for the development of vehicular network systems because it does not require experiment setup in the real world and can be conducted at low costs. However, the existing network emulators focus on link emulation. In other words, it emulates the throughput and packet loss in vehicular networks and does not allow the IEEE 802.11 protocol implementation to operate with the network applications. The existing network emulators incompletely meet the needs for the development of vehicular networks. For this reason, this study proposes a novel wireless LAN emulator with wireless network tap devices that enables validating the operation of both network applications and IEEE 802.11 protocol stacks for Linux systems with the support of flexible radio propagation and mobility simulation using a network simulator.

The second contribution is the first empirical investigation of the initial link setup time and bulk data transmission in inter-vehicular communications. This study reveals initial link setup time reduction by Fast Initial Link Setup (FILS) in IEEE 802.11-based vehicular networks based empirical measurements. In an inter-vehicular communication between vehicles passing each other with high speeds, the vehicles have only a short time to establish a link and trasfer bulk data. To increase the amount of transmitted data size between the vehicles, the overhead for link establishment should be as short as possible. FILS can complete IEEE 802.11 link establishment in 100ms and has the potential to shorten the link establishment overhead in the inter-vehicular communication, but no paper investigates the FILS performance in the inter-vehicular communication in the real world as far as the author know. Therefore, this study investigates and shows the field performance of FILS in the inter-vehicular communication in the real world. In addition, this study shows the field performance of bulk data transmission between vehicles passing each other with high relative speed. This study compares the field performance of FILS and Wi-Fi Protected Access with Protected Extensible Authentication Protocol (WPA-PEAP) in inter-vehicular communication and reveals that FILS can shorten the practical initial link setup time ten times than WPA-PEAP and allows vehicles to transfer 75MB while passing each other with high relative speed.

The remainder of this dissertation is structured as follows. Chapter 2 describes the wireless LAN emulator with wireless network tap devices (WiNE-Tap). WiNE-Tap enables a wireless LAN emulation that the implementation of IEEE 802.11 protocol stacks of Linux works cooperatively with the behavior of the physical layer, such as radio propagation and network nodes' mobility, which is simulated by a network simulator. Chapter 3 describes the performance evaluations of initial link setup time and bulk data transmission in intermittent inter-vehicular communications based on empirical measurements. Chapter 3 also shows the field performance comparison of FILS and WPA-PEAP and the FILS effectiveness on improving the inter-vehicular bulk data transmission. Chapter 4 concludes this dissertation and discuss the utilization of the contributions of this dissertation: the wireless network emulator and the field measurement results of bulk data transmission in inter-vehicular communications, and the prospects of vehicular networks.

Chapter 2 Wireless LAN emulation with wireless network tap devices

This chapter describes a novel wireless LAN emulator with wireless network tap devices. The wireless network emulator can reproduce the behavior of Linux IEEE 802.11 protocol stacks and network applications with their actual implementation as they work in the real IEEE 802.11 network. This study shows the architecture and performance evaluations of the wireless network emulator and some techniques to improve the performance of the emulator based on the experiment results. This chapter is written based on the author's paper [35].

2.1 Introduction

Operation validation using the practical implementation is a vital process for developing a vehicular network system to avoid any accidents due to the malfunction of the system. Network simulation is often used to validate the performance of vehicular network protocols or applications, and several vehicular network simulation environments and models have been proposed in[36, 37, 38, 39]. For example, Veins[40] and VENTOS[39] are network simulators consisting of OMNeT++ network simulator and a traffic simulator SUMO (Simulation of Urban Mobility)[36]. Based on the real-world GIS (Geographic Information System) map and propagation models of the network simulator, veins simulates the behavior of a vehicular network system with realistic traffic traces simulated by SUMO.

On the other hand, since network simulation abstracts the behavior of the entire vehicular network system with simulation models, it does not minutely reproduce the behavior of the vehicular network system with implementation to the actual onboard units. The behavior of the vehicular network system in the real road environment depends on the implementation to onboard units and other actual network equipment of the vehicular network system and communication environments in a field in where the system is used. Therefore, it is essential to validate the operation of the vehicular network system with its implementation.

There are two ways to conduct the operation validation using implementation: field testing and network emulation. Field testing reveals the practical performance of the vehicular network system in the real field. Meanwhile, network emulation allows network applications to work in a virtual network in which a network simulator reproduces the radio propagation and the mobility of network nodes and signal transmission and reception by network equipment. Network emulation is more convenient than field testing in the early phase of developing the vehicular network systems because it can be performed without the preparation of network and vehicle equipment necessary for building the network in the real world.

Some existing network simulators, such as ns-3[41], Mininet[42], Scenargie[43], and EXata[44], support network emulation. These network simulators capture packet flows exchanged between network applications via virtual network devices such as a TUN/TAP device[45] and apply transmission delays insertion, bandwidth limitation, and packet loss based on radio propagation and mobility models. However, the existing network emulators have some restrictions listed below.

- They incompletely emulate the control of wireless network devices by operating systems because their network emulation mechanism applies wired network virtualization and focuses on link emulation between network applications.
- A network simulator and network applications work on the same host machine and allow the high load computing load to the host machine.

- They depend on some specific network simulators' architecture and do not allow users of the emulators to select a network simulator they want to use.

The existing network emulators enable link emulation using network applications used in the real environment, considering the radio propagation and network node mobility but do not focus on reproducing the behavior of wireless network protocols such as Linux IEEE 802.11 implementation. For example, ETSI DCC changes transmission power levels based on received signal strengths to avoid interference. However, since the existing network emulators do not allow the network protocol stacks such as Linux IEEE 802.11 implementation to work with the network applications in the virtual network, they do not emulate the behavior of these vehicular network protocols.

Almost all existing network emulators also have some inconvenience limitations to be used for vehicular network systems. The first limitation is that the existing network emulators can hardly be scalable because network applications and a network simulator work on the same machine. The system load concentrates on the machine on the current network emulators, such as ns-3's DCE[46] and Mininet-WiFi[47]. The second limitation is that existing network emulators are designed for a specific network simulator and depend on their limitations. For instance, Mininet-WiFi[47] simulates transmission delays over Wireless LAN by the `tc` command[48], a traffic control command for Linux, or wmediumd[49], a lightweight frame loss/delay simulator. However, `tc` command does not simulate radio propagation and network nodes' mobility. Wmediumd supports few radio propagation models, such as free space propagation, and cannot emulate complicated IEEE 802.11 networks such as vehicular networks. To remove the limitations, vehicular network emulatioin should be meet the items listed below.

- Network simulators supporting IEEE 802.11 emulation should also support importing/exporting IEEE 802.11 frames and IEEE 802.11-related control parameters, such as transmission power and received signal strength.

- Mechanisms that enable an operating system and a network simulator to exchange IEEE 802.11 frames and IEEE 802.11-related control parameters should be implemented. Concretely, they require interfaces to import/export the IEEE 802.11 frames and IEEE 802.11-related control parameters and middleware that converts message formats if needed and exchange the frames and parameters between the interfaces.

This study presents a new wireless network emulator, WiNE-Tap, covering these solutions. We implemented the mechanisms that convert message formats and enable the Linux kernel and a network simulator to exchange IEEE 802.11 frames and control parameters for manipulating IEEE 802.11 devices. This chapter also presents the first implementation of functions importing/exporting IEEE 802.11 frames and IEEE 802.11-related control parameters to a network simulator, Scenargie. In the following sections, the architecture and performance evaluation of WiNE-Tap are descibed after introducing the related work.

2.2 Related work

This section explains the mechanisms of the existing network emulators, focusing on IEEE 802.11 network emulation. Most of the existing network emulators utilize virtualization technologies such as hypervisors and containers. This section first introduces the virtualization technologies and classifies the existing network emulators into several categories based on the differences between virtualization technologies.

2.2.1 Virtualization technologies relative to realizing IEEE 802.11 network emulation

Virtualization has several meanings in the computing world and can be roughly defined as abstracting the hardware and software resources of a computer, for example, CPU, memory, storage devices, network devices, file systems. The virtualization mechanisms can be categorized according to which computer resources are abstracted and what abstracted resource is used. This chapter focuses on two of the categories used in network emulation: machine virtualization and network virtualization.

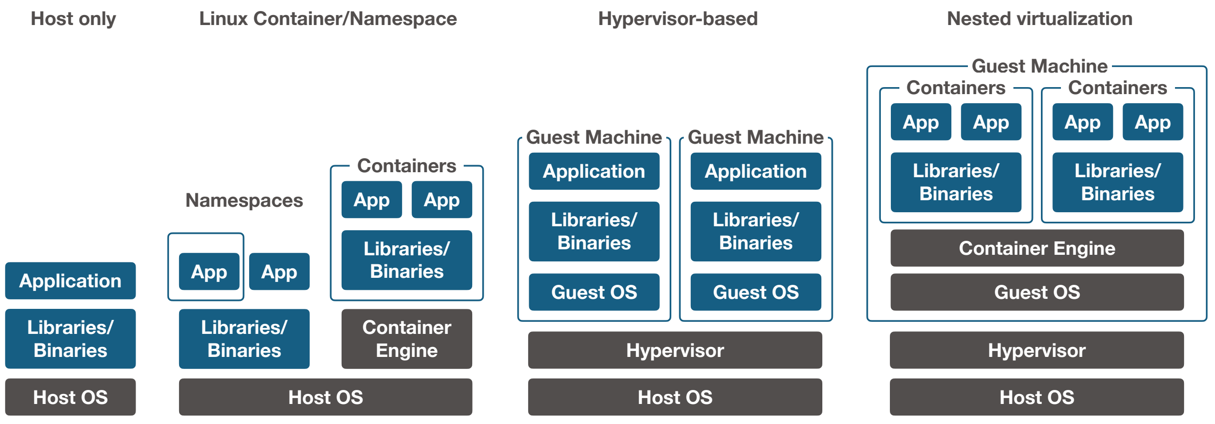

Figure 2.1: Differences among machine virtualization technologies

■ Machine virtualization

Machine virtualization is a technique that abstracts the whole single computer and creates a virtual machine (VM) on a real computer. The virtual machine works as a real computer without any difference, except it does not have the hardware. There are two systems to realize machine virtualization: hypervisor-based machine virtualization and container-based virtualization. Figure 2.1 shows the differences between machine virtualization technologies.

The hypervisor is a software program that allows operating systems (OSs) and user applications to work on virtual machines and reproduces the operation of the hardware of a computer, for example, CPUs, chipsets on a motherboard, storage devices, network devices by using virtual devices. A virtual device is a program that returns the output values as those returned from the actual hardware against the input value from the operating system. The hypervisor-based machine virtualization creates the virtual machine that minutely reproduces a computer.

Container-based machine virtualization logically divides computer resources and enables the divided resources to be assigned to specific processes. For example, Docker[50], which is one of the container-based virtualization platforms, assigns the logically divided computer resources to instances. An instance is a set of software packages that include all software components required to work as a single computer. The instance behaves like a single computer on a real computer.

Hypervisor-based machine virtualization is different from container-based machine virtualization at the point that the former reproduces the behavior of hardware components by software, but the latter just assigns divided computer resources to the instances. In other words, the container-based machine virtualization executes the software components of an operating system on another operating system, and the instances and the host machine share computing resources. While the hypervisor-based machine virtualization emulates the operation of the CPU architecture differently from the host machine, the container-based machine virtualization does not emulate such operation. For example, QEMU[51], a generic and open source machine emulator and virtualizer, works as a hypervisor and allows an ARM-based virtual machine to run on an AMD64-based real machine.

■ Network virtualization

Network virtualization is a technique to construct a virtual network and allows user applications to perform in the virtual network as if they work on a real network. In particular, network virtualization is used to reproduce and examine the performance of the network applications or protocol stacks based on a specific network scenario. For example, the Linux traffic controller (tc)[48], netem[52], Dummynet[53], KauNet[54] enable to classify packet flows and restrict bandwidth and packet loss ratio of the classified packet flow based on the preconfigured values or the trace data of the actual bandwidth and packet loss ratio on a real network. The network emulation methodology that conducts bandwidth restriction and delays insertion to packet flows is called link emulation, and the network emulators, which abstract the network layer or below and perform link emulation, are called link emulators[53]. The basic idea of link emulators, such as link classification and bandwidth limitation, can be applied to IEEE 802.11 network emulation. For example, KauNet[54], an extension of Dummynet implementation, allows a wireless network emulator W-NINE[55] to control classified packet flows based on the trace data calculated from the radio propagation, mobility, and IEEE 802.11b/g protocol models.

■ Virtualization technologies for network devices

The network simulators described in the previous section capture packet flows through virtual network devices such as TUN/TAP device[45], Virtual Ethernet device (namely veth device)[56], and virtual interface (namely vif). Although the word, network interface, can be referred to as both the network device such as an Ethernet card or a software entry point in charge of sending and receiving data packets, this study defines a network device and a network interface as hardware able to exchange data with other hosts over a network and a software entry point between network applications and the network protocol stacks, respectively. Following the definition of the network device and interfaces, the virtual network device can be defined as the program that receives packets via the network interface associated with itself and forwards the packets to a user application instead of a network.

A TUN/TAP device provides a virtual network interface and a file descriptor to user applications and transfers captured IP packet flows or Ethernet frame flows from the interface or the TCP/IP stacks to the file descriptor. It can also transfer data sent through the file descriptor to the virtual network interfaces, in which the data are encapsulated into packets or frames in the network protocol stacks before they reach the virtual network interfaces. The TUN/TAP device is available on various operating systems such as Linux, FreeBSD, Windows, and macOS.

A virtual ethernet device is a virtual network device for Linux systems and functions similar to the TUN/TAP device, but it provides user applications with a pair of two virtual network interfaces instead of providing a file descriptor. The virtual interfaces are connected through a Linux Netlink socket[57], which is one of socket APIs and is used for inter-process communication between the user space and the kernel space. Thus, user applications connected via the `veth` device perform as they communicate through network interfaces.

The virtual interface (vif) is a virtual network interface associated with another network interface such as a bridge interface, and a network interface is associated with a network device. The virtual interface is used as a wrapper interface of the real network interface to set different network configuration parameters to the real one simultaneously.

Some researchers proposed a network emulator with unique virtual devices. For example, EMPOWER [58] is a network emulator using a virtual device that works between the IP stack and the device driver. EMPOWER's virtual device captures IP packet flows and applies for the bandwidth limitation, delay insertion, and packet loss before forwarding the captured IP packet flows to the device drivers.

EtsiNet [59] uses a network tunneling interface that captures IP packet flows similar to TUN/TAP devices and forward the captured packet flows to the simulation engine, which conducts link emulation for wired networks. EtsiNet's simulation engines reproduce the behavior of network protocol stacks between the physical and Media Access Control (MAC) layers instead of the native protocol stacks embedded in the operating system.

2.2.2 Hypervisor-based network emulation

Network emulators with hypervisor-based machine virtualization have been proposed in [60], [61], [62]. In the hypervisor-based network emulation, a virtual machine performs as a network node. The virtual machine executes network applications and network protocol stacks as these programs work in the real environment.

Kawai's network emulator[60] utilizes the kernel-based virtual machine (KVM). The KVM is a hypervisor implemented in the Linux kernel and works as Linux kernel modules. In Kawai's network emulator, the KVMs are connected to a discrete-event network simulator, Scenargie[43], through inter-process communication (IPC) and communicate with each other via Scenargie. Scenargie supports IEEE 802.11 network simulation and restricts the data rate and packet loss ratio of data transmission between the KVMs based on its IEEE 802.11 PHY/MAC and mobility models. Kawai et al. also have implemented a virtual wireless LAN device in the KVMs, which works as a Qualcomm Atheros AR9160 Wireless LAN card operates. The virtual wireless LAN device enables the Linux Atheros device driver, `ath9k`, to operate on the KVM. Thus, Kawai's network emulator emulates the behavior of network user applications, network protocols stacks, and the Atheros wireless LAN device driver.

Akashi's network emulator[61] works on a computer cluster and executes multiple virtual machines on the computer cluster simultaneously. The virtual machines connect through the backbone network that connects the cluster machines. The data traffic among the virtual machines are controlled by the program that calculates the bandwidth, packet loss ratio, and connectivity among the virtual machines in the virtual network based on (pre-)configured network scenario. Akashi's network emulator enables a large-scale wireless network emulation with sufficient computing resources from the computer cluster. While Kawai's network emulator works on a single computer and has limited computer resources, Akashi's network emulator can utilize more computing resources from the computer cluster and enables a large-scale network emulation.

Staub et al. [63] use Xen hypervisor to emulate virtual wireless mesh network nodes. The virtual mesh network nodes are connected to a network simulator, OMNeT++, and the network simulator calculates radio propagation between the virtual network nodes.

The network emulator proposed by Luca et al. uses Java VM instead of the KVM[64]. Luca et al. use Java VMs with network applications and TCP stacks written in Java instead of the KVMs and connect the Java VMs over the TUN/TAP devices on the host machine. The network emulator lacks the reality of network emulation due to the re-implemented TCP/IP stack in Java, but it is an example of network emulators using virtual machines.

As of 2021, since various cloud platforms that support hypervisor-based machine virtualization are provided by Amazon[65], Microsoft[66], and Google[67], some researchers have extended network emulation environments to the cloud platforms. For example, Michael et al. [62] implemented a network virtualization environment in Google Cloud Platform (GCP)[67] and reported that their network emulation environment scales up a virtual network to 10,000 virtual machines.

2.2.3 Container-based/Namespace-based network emulation

The container-based network emulation uses instances as network nodes in a virtual network. Since the instances consume less computing resources than virtual machines, the container-based network emulation has good scalability. Most of the container-based network emulators connect the instances using the virtual network devices and network bridge interfaces such as the Linux bridge interface[68] and Open vSwitch[69]. The network bridge interface emulates a Layer 2 (L2) Ethernet switch and allows multiple network interfaces to communicate frames with each other and relays packet flows among the virtual network devices in container-based network emulation environments. For example, DOCUEMU [70] is a network emulator using Linux containers connecting via bridge interfaces on a single Linux machine.

For example, the network emulator proposed by Handigol et al.[71] or Yan et al.[72] use Open vSwitch to connect containers and conduct link emulation by controlling the packet flows via Open vSwitch. Open vSwitch is a distributed virtual network switch and forms a virtual network switch over multiple network hosts. Open vSwitch has the packet flow control function and performs as a link emulator.

The Extendable Mobile Ad-hoc Network Emulator (EMANE) is a wireless network emulation framework and supports containers[73]. EMANE consists of several subsystems which enable real-time wireless link emulation with the simulation models of radio propagation, mobility, and wireless network protocol stacks such as IEEE 802.11 and Long Term Evolution (LTE) and 4G networks. EMANE provides the user spaces with TUN/TAP devices as the interfaces between user applications and the EMANE subsystems. EMANE allows user applications and instances to be network nodes in the emulated network and communicate via the TUN/TAP device and applies link emulation to the packet flows among the network nodes.

2.2.4 Network emulators using nested virtualization

Nested virtualization is a technique that allows instances to work in a virtual machine. Nested virtualization is more scalable than hypervisor-based virtualization in network emulation because the instances consume less computing resources than virtual machines when they emulate the same number of network nodes. For example, the network emulator proposed by Grau et al.[74] is constructed with Linux containers on virtual machines running on the Xen hypervisor[75]. Grau's network emulator aims to provide highly accurate time synchronization among the instances by adjusting a virtual time clock of the Xen hypervisor.

2.2.5 Network emulators based on hardware-based channel emulation

Simulation models incorrectly calculate radio propagation fluctuations in vehicular networks because they cannot consider indefinite factors from natural environments and hardware characterizations such as RF modulation. For emulating the fluctuation of radio propagation, some network emulators have hardware-based channel emulation mechanisms.

The network emulators proposed in [76, 77, 78] emulate radio frequency (RF) behaviors with Field Programmable Gate Array (FPGA) based channel emulation board. These network emulators allow the FPGA board with a channel emulator using real RF signals emitted from network cards to emulate radio propagation delays instead of a network simulator. Using the FPGA board helps to reproduce the behavior of radio signal processing. FPGA reduces the calculation overhead on radio propagation simulation and realizes better performance than a network simulator. On the other hand, these network emulators require users to prepare FPGA devices and program the FPGA boards. Thus, these network emulators force users to spend additional time on programming.

Ghiaasi et al. [79] propose a vehicular network emulator collaborating with Software Defined Radio (SDR) devices to emulate IEEE 802.11p radio propagation. The SDR device is connected to the real onboard units using coaxial cables and generates signals based on the control by the host computer. Using the real onboard units, the network emulator can reproduce the behavior of the onboard units when they communicate over the radio with the actual radio interface. However, it is less scalable because it requires many onboard units when emulating a large-scale network.

Network emulation with hardware-based channel emulation has better strictness to evaluation results than others, but it forces to take extra time to construct the network emulation testbed and conduct complicated setups. In addition, FPGA-based channel emulation makes it difficult to simulate large-scale network emulation considering radio propagation and mobility of many vehicles.

2.2.6 Virtual network devices designed for IEEE 802.11 network emulation

Although almost all the existing network emulators utilize virtual Ethernet devices to capture data packet flows, two virtual network devices are designed for IEEE 802.11 networks. Weighgartner et al. developed a virtual wireless network device with functions similar to the TUN/TAP device in [80]. Weighgartner's virtual network device works with Wireless Extension, an IEEE 802.11 protocol implementation for Linux systems and supports IEEE 802.11n and older IEEE 802.11 protocols. It provides a user application with a virtual wireless network interface and exports IEEE 802.11 frames to another user application. The Wireless Extension generates the IEEE 802.11 frames from data packets via the virtual wireless network interface. Weighgartner's device enables wireless LAN emulation using real IEEE 802.11 frames. The Wireless Extension was replaced with the Linux wireless subsystem, a newer IEEE 802.11 implementation than the Wireless Extension. The Wireless Extension does not support the latest IEEE 802.11 standards such as IEEE 802.11ac or newer protocols and is no longer updated. Thus, Weigngartner's virtual wireless network device does not also support the latest IEEE 802.11 standards.

Another virtual wireless network device is `mac80211_hwsim`, a virtual SoftMAC driver. `mac80211_hwsim` aims to be used for debugging the implementation of the IEEE 802.11 protocol stacks for Linux systems and supports the Linux wireless subsystem. Therefore, it has limitations inconvenient to be used for network emulation. The SoftMAC driver notifies a wireless LAN device of configuration parameters configured by network applications such as transmission frequency and transmission power. For example, if a user configures the channel number using Linux wireless LAN configuration utility (`iw`), the SoftMAC driver configures the wireless network device with the changed parameters. Similar to Weighgartner's virtual wireless network device, `mac80211_hwsim` sends and receives IEEE 802.11 frames to/from user applications but does not notify user applications of the changed parameters. Thus, the user application cannot realize that the IEEE 802.11 protocol stacks configure the wireless network device through `mac80211_hwsim`.

Some network emulators use `mac80211_hwsim`[81]. The network emulators using `mac80211_hwsim` collaborate with other network simulators supporting link emulation for IEEE 802.11 networks because `mac80211_hwsim` does not have the link emulation function.

For example, the network emulator proposed by Gilles et al.[81] uses `mac80211_hwsim` to connect network applications isolated with Linux namespaces. Gilles's network emulator connects the virtual wireless network devices emulated by `mac80211_hwsim` using the bridge interface, but it does not have interfaces to provide user applications with the IEEE 802.11 frames and the configuration parameters for wireless network devices. Thus, it is difficult to collaborate with network simulators running on the userspace such as ns-3 and Scenargie.

Another example of such a network emulator is `Wmediumd`[49]. `Wmediumed` is a link emulator working with `mac80211_hwsim`. `Wmediumd` captures IEEE 802.11 frames from `mac80211_hwsim` and inserts the propagation delays to the frame flow based on radio propagation model such as the free space model before repeating the frame flow back to `mac80211_hwsim`. `Wmediumd` is supported by the Linux kernel because `mac80211_hwsim` has the interface to connect to `Wmediumd`. However, it supports a few radio propagation models and no dynamic mobility simulation. `Wmediumd` calculates the propagation delays if the mobility trace files of network nodes are given before link emulation.

2.2.7 Problems of IEEE 802.11 network emulation

The problems of the existing IEEE 802.11 network emulation can be listed below.

- Insufficient interoperability between the IEEE 802.11 protocol stacks in the kernel and the network simulator
- Trade-off between the scalability of the network emulators and the strictness of the emulation results
- Limited collaboration with a network simulator

Hypervisor-based network emulation enables to emulate the behavior of a network system more strictly than other methods because the implementation of network protocol stacks and network device drivers work on a virtual network. However, it emulates network nodes using hypervisor-based machine virtualization and consumes more computing resources as the number of network nodes increases. Remarkably, the network emulators like Kawai's network emulator execute virtual machines and a network simulator on a single machine and allow the computing load to concentrate on the host machine. Thus, the high computing load due to emulating many network nodes takes the correctness of the results of network emulation from these network emulators. In other words, hypervisor-based network emulation benefits validating the correctness or finding the software bugs of these implementations.

Network emulators built on a single machine has less scalability than network emulators dispersively built on multiple machines because the former allows the computing load to concentrate on the machine.

Using cloud computing resources benefits scaling up the network emulation and reducing the computing load for executing many virtual machines because it is practical to use multiple host machines to execute many virtual machines. However, it imposes complicated configuration management for constructing computing clusters and network emulation environments on its users. Some researchers suggest the problem and proposed configuration tools to reduce the loads for configuration management. Major configuration utilities for constructing cloud computing environments such as Kubernetes(k8s)[82], Ansible[83], Chef[84], and Vagrant[85] possibly helps to mitigate the burden on the configuration management. However, the complicated configuration management is contrary to the needs of most users of network emulation, which they aim to spend time to test their network applications, not to build the network emulation environment.

The existing virtual network devices have sufficient functionality to emulate link emulation for IEEE 802.11 networks. However, they are insufficient to examine the IEEE 802.11 protocol stacks on network emulation. Some virtual network devices can provide user applications with IEEE 802.11 frames the same as those exchanged between the wireless network hosts in a real wireless network but do not provide configuration parameters when the IEEE 802.11 protocol stacks change them. The network emulators using `mac80211_hwsim` enable wireless network emulation using the Linux wireless subsystem but do not have the function to inform the wireless LAN configuration of user applications such as network simulators.

The discussion above indicates that the strictness and scalability in network emulation have the relation of trade-off. In addition, most of the existing network emulators focus on expanding the scale of a virtual network and emulating the performance of application data transmission. This is in contrast to the requirements of developing wireless communication devices for vehicular networks. As mentioned in Chapter 1, since the development of onboard units requires developers to observe the requirements[34] of the onboard units, the network emulation environment that meets the requirements of developing vehicular network systems is indispensable for the developers and researchers to encourage their works.

2.2.8 Requirements for emulating IEEE 802.11 network systems.

There are two requirements that the existing network emulators do not meet for IEEE 802.11 network emulation.

- Expanding to multiple physical machines
- Interoperability with various network simulators

The existing network emulators focus on link emulation only. An IEEE 802.11 device controls not only data transmission but also the IEEE 802.11 MAC operations; for example, it configures channels and adjusts transmission power depending on the radio link-state, such as received signal strength. In the existing network emulation, the IEEE 802.11 MAC operations are fully simulated by a network simulator, and the IEEE 802.11 protocol stacks implemented in the kernel of an operating system do not work in network emulation. Therefore, it is difficult to link emulation for IEEE 802.11 networks is based on radio propagation, mobility, and IEEE 802.11 protocol models.

2.3 System architecture

This section proposes the wireless LAN emulator with wireless network tap devices (WiNE-Tap), a novel wireless network emulator fully supporting the IEEE 802.11 protocol stacks implemented in the Linux kernel, and describes the architecture and mechanisms of WiNE-Tap. The source code of WiNE-Tap is available online at the Github page at `https://github.com/ishilab/wine-tap`.

The wireless LAN emulator with wireless network tap devices (WiNE-Tap) consists of three software components: wireless network tap device (`wtap80211`) that is a virtual network device supporting IEEE 802.11 networks, a user daemon process that assists the functions of `wtap80211` (hereafter, `wtap80211-daemon`), and the software library that provides user applications with interfaces to connect to `wtap80211-daemon` (hereafter, `Libwinetap`). In addition, `wtap80211` and `wtap80211-daemon` consist of the wireless network emulation framework. The feature of WiNE-Tap is that it enables emulating data transmission over IEEE 802.11 networks with actual IEEE 802.11 frames and the configuration parameters held in the Linux wireless subsystem. WiNE-Tap enables the Linux wireless subsystem and user applications such as network simulators to exchange the IEEE 802.11 frames and the configuration parameters in real-time and allows the Linux wireless subsystem to operate following the radio propagation and mobility of network nodes in a virtual network simulated by the network simulator. Therefore, WiNE-Tap enables emulating the operation of the IEEE 802.11 protocol stacks for Linux systems in the virtual network.

The wireless network emulation framework helps user applications to know the behavior of the IEEE 802.11 protocol stacks, which are usually invisible to the user applications. The `Libwinetap` helps user applications such as network simulators decode the messages from the wireless network emulation framework, including the IEEE 802.11 frames and the configuration parameters captured by `wtap80211`.

Since the wireless network emulation framework utilizes the APIs of the IEEE 802.11 protocol stacks for Linux, the following sections explain the mechanisms of the Linux wireless subsystem and the wireless network emulation framework. Then they describe the architecture of WiNE-Tap.

2.3.1 Wireless network emulation framework

The wireless network emulation framework is developed by the author and consists of the wireless network tap device (wtap80211) and the user daemon process (wtap80211-daemon) that assists encoding/decoding messages that wtap80211 sends or receives. Fig-

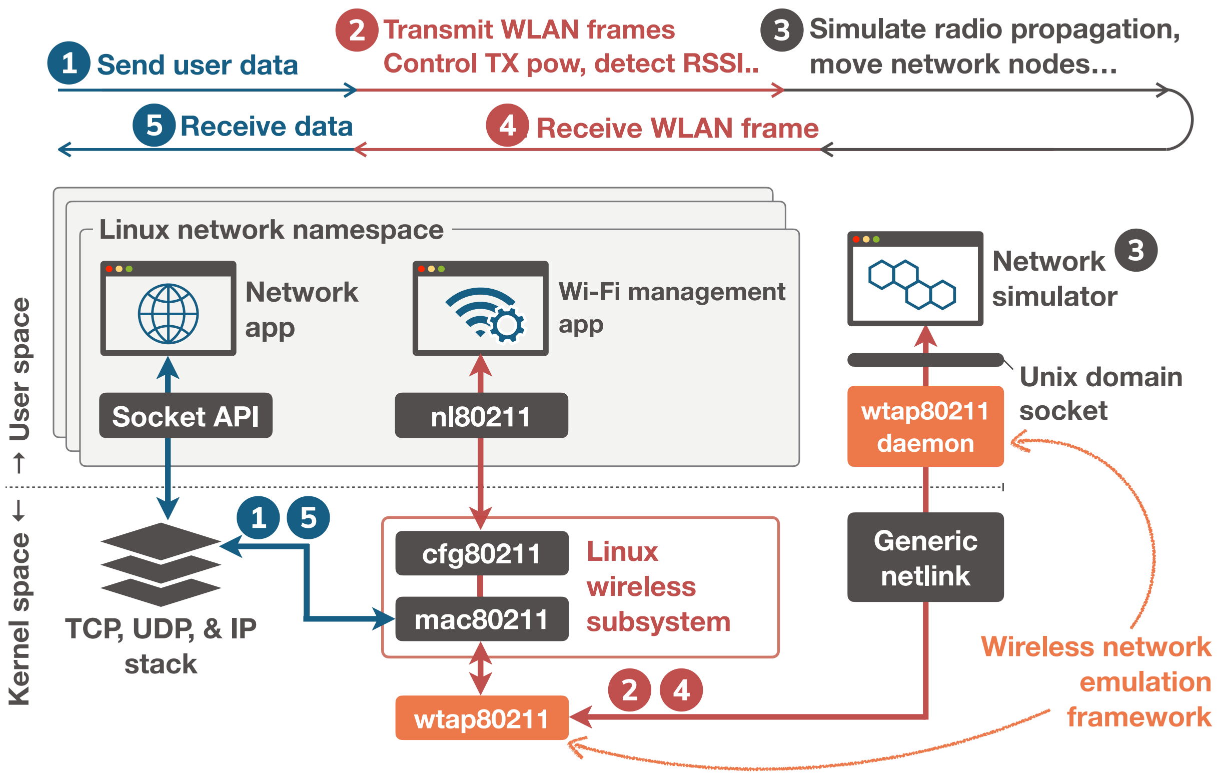

Figure 2.2: Wireless Network Emulation Framework

ure 2.2 shows the architecture of the wireless network emulation framework.

■ Wireless network tap devices

The wireless network tap device (wtap80211) is a virtual IEEE 802.11 network device. `wtap80211` captures IEEE 802.11 frames such as association, authentication, and data frames, which the IEEE 802.11 protocol stacks in the Linux kernel generate and sends to IEEE 802.11 network devices, and transfers the captured IEEE 802.11 frames to user applications via Linux Netlink sockets. In addition, `wtap80211` also monitors the IEEE 802.11 network device configuration and notifies the configuration parameters of the IEEE 802.11 network devices to the user applications when the IEEE 802.11 protocol stacks change the configuration parameters. The function to transfer the configuration parameters to the user applications is the original function of `wtap80211`, and other virtual network devices do not have such function.

The wireless network tap device is implemented as a virtual SoftMAC driver. Although the real SoftMAC driver informs the IEEE 802.11 protocol stacks of the device configuration information read from the firmware on the IEEE 802.11 network device and allows the IEEE 802.11 protocol stacks to recognize and control the device, `wtap80211` notifies the IEEE 802.11 protocol stacks of pseudo IEEE 802.11 network device configuration information shown in Table 2.1 and makes the IEEE 802.11 protocol stacks recognize that the IEEE 802.11 network device is attached to the Linux system even though the IEEE 802.11 device does not actually exist.

Table 2.1: Pseudo wireless LAN device configuration information of `wtap80211`

Name	Specification
Kernel version	Linux 3.19, 4.0.x ˜ 4.4.x
Standards	IEEE 802.11a/b/g/n/ac, IEEE 802.11p/s/p (limited support)
Operation modes	Managed (STA), Master (AP), IBSS (Ad-hoc),
	Monitor, Mesh, OCB (Out context of BSS)

The IEEE 802.11 protocol stacks for Linux systems comprises two Linux kernel modules, `cfg80211`[86] and `mac80211`[87]. `cfg80211`mediates the network application in the user space such as `hostapd` and `wpa_supplicant` and either `mac80211` or the IEEE 802.11 network device driver depending on which the IEEE 802.11 device driver is the FullMAC driver or the SoftMAC driver[88]. If the IEEE 802.11 network device driver is the FullMAC driver, `cfg80211` directly accesses the IEEE 802.11 network device via the FullMAC driver. If not, `cfg80211` indirectly accesses the IEEE 802.11 network device through `mac80211`.

The SoftMAC and FullMAC drivers are different types of implementation methods of IEEE 802.11 network device drivers for Linux systems. The FullMAC driver performs as the IEEE 802.11 Mac Access Control Sublayer Management Entity (IEEE 802.11 MLME). In contrast, the SoftMAC driver does not implement the IEEE 802.11 MLME and allows the IEEE 802.11 network devices to process the tasks of the IEEE 802.11 MLME. In Linux systems, `mac80211` takes the IEEE 802.11 MLME. `cfg80211` registers the IEEE 802.11 network devices to the IEEE 802.11 protocol stacks in the Linux kernel and manages the channel configuration of the IEEE 802.11 network devices based on the regulatory domain database. `cfg80211` is also responsible to manages the power control of the IEEE 802.11 devices. For this reason, `mac80211` and `cfg80211` are closely related to the behavior of the IEEE 802.11 network systems.

For example, ETSI Decentralized Congestion Control (DCC) adjusts transmission power based on the change of received signal strength to avoid signal interference[89]. Since `cfg80211` controls the transmission power of IEEE 802.11 network devices in Linux systems, the IEEE 802.11-based vehicular network system emulation requires `cfg80211` to operate with network applications during emulation and makes it follow the simulated behavior of radio propagation and mobility among IEEE 802.11 network nodes in the virtual network.

The conventional virtual network devices such as TUN/TAP devices are based on Ethernet device drivers and do not work with the IEEE 802.11 protocol stacks for Linux systems. Therefore, they do not have mechanisms to provide the received signal strength simulated in the virtual network to the IEEE 802.11 protocol stacks and do not have a mechanism to inform the behavior of the IEEE 802.11 protocol stacks such as transmission power changes to the network simulator. For this reason, it is difficult for the existing network emulators to emulate the behavior of the IEEE 802.11 protocol stacks. In contrast, `wtap80211` is based on the IEEE 802.11 network device drivers and has mechanisms to provide the received signal strength, transmission power, and other configuration values to the IEEE 802.11 protocol stack and the network simulator each other. Table 2.2 shows the examples of parameters that `wtap80211` captures and provides to the IEEE 802.11 protocol stacks and a network simulator.

As shown in Figure 2.2, the IEEE 802.11 protocol stacks for Linux work independently from other network protocol stacks such as TCP/IP. The data flow and configuration flow are separated at the IEEE 802.11 protocol stacks. When receiving an IEEE 802.11 data frame, the IEEE 802.11 protocol stacks decapsulate an IP packet from the received frame and turn the IP packet over to the TCP/IP stack. Meanwhile, when the IEEE 802.11 protocol stacks receive an IEEE 802.11 management/control frame, the IEEE 802.11 protocol stacks retrieve the parameters included in the received management/control frame and notifies the parameters to particular user applications such as hostapd[90] an IEEE 802.11 access point implementation for Linux, or wpa_supplicant[91], Linux WPA supplicant implementation.

Table 2.2: Examples of control parameters that `wtap80211` captures and provides to the IEEE 802.11 protocol stacks and a network simulator

Parameter by categories	Type	Description
Device configuration		
SIGNAL_DBM	bool	Signal values are in dBm if true.
AMPDU_AGGREGATION	bool	AMPDU aggregation is enabled if true.
QUEUE_CONTROL	bool	TX queue is controlled by software if true.
Hardware state information		
IDLE	bool	the device is in idle state if true.
POWER	bool	the device is turned on by software if true.
MONITOR	bool	the device operates in monitor mode if true.
Channel configuration		
BAND	int8	Current channel band number
CENTER_FREQ	int32	Center frequency in MHz
MAX_POWER	int8	maximum transmission power in dBm
MAX_ANTENNA_GAIN	int8	maximum antenna gain in dBi
BSS information		
SSID	char[]	Extended SSID
IBSS	bool	the device is in an IBSS network if true.
BEACOM	bool	beacon parameter is changed if true.
OCB	bool	the device is in an OCB network if true.

2.3.2 wtap80211-daemon

`wtap80211-daemon` is a user daemon process to assist communication between `wtap80211` and user applications by encoding and decoding messages between them. Since the messages from `wtap80211`are packed in Generic Netlink messages, the user applications that communicate with `wtap80211` are responsible for encoding and decoding the messages from `wtap80211`. The Generic Netlink message can be decoded using `libnl`, a static library containing convenient wrapper functions for the native Netlink APIs. However, the Generic Netlink APIs requires user applications to define the payload format with the data structures specified by the Generic Netlink APIs and are more complicated than other network socket APIs such as TCP/IP sockets, and it possibly enforces on developers to spend more time to implement the message encoding/decoding function to their user applications. Therefore, `wtap80211-daemon` decodes the message from `wtap80211` instead of user applications.

The wireless network emulation framework makes the behavior of the Linux wireless subsystem visible to the user applications but works on a single machine only and cannot be scalable to multiple machines simultaneously. Thus, the author extended the functionality of the wireless network emulation framework and newly developed a wireless LAN emulator using the wireless network emulation framework (WiNE-Tap).

2.3.3 Mechanisms of WiNE-Tap

The wireless LAN emulator with wireless network tap devices (WiNE-Tap) is an IEEE 802.11 network emulator for Linux systems and can consist of single or multiple machines. Figure 2.3 shows the architecture of WiNE-Tap. WiNE-Tap extends the functions of the wireless network emulation framework and makes the behavior of the Linux wireless subsystem visible to the user applications running on multiple machines by using the extension of `wtap80211-daemon` and an additional software component, `Libwinetap`. Concretely, WiNE-Tap enhances the message forwarding function of `wtap80211-daemon` to transfer the decoded message from `wtap80211` to other machines connected via Ethernet. `Libwinetap` provides APIs to network simulators to decode the messages from `wtap80211` and encode the data packet with the simulated received signal strengths and other parameters to the message format wtap80211‘ receives and decodes.

`wtap80211-daemon` connecting over Ethernet communicates the messages from `wtap80211` with jumbo frames. The maximum length of an IEEE 802.11 frame is 2346 bytes or much longer when multiple IEEE 802.11 frames are aggregated. However, the maximum length of an Ethernet frame is 1500 bytes and smaller than the maximum length of the IEEE 802.11 frame, and the IEEE 802.11 frame should be transferred after being divided into multiple Ethernet frames. The division and unity of IEEE 802.11 frames during the emu-

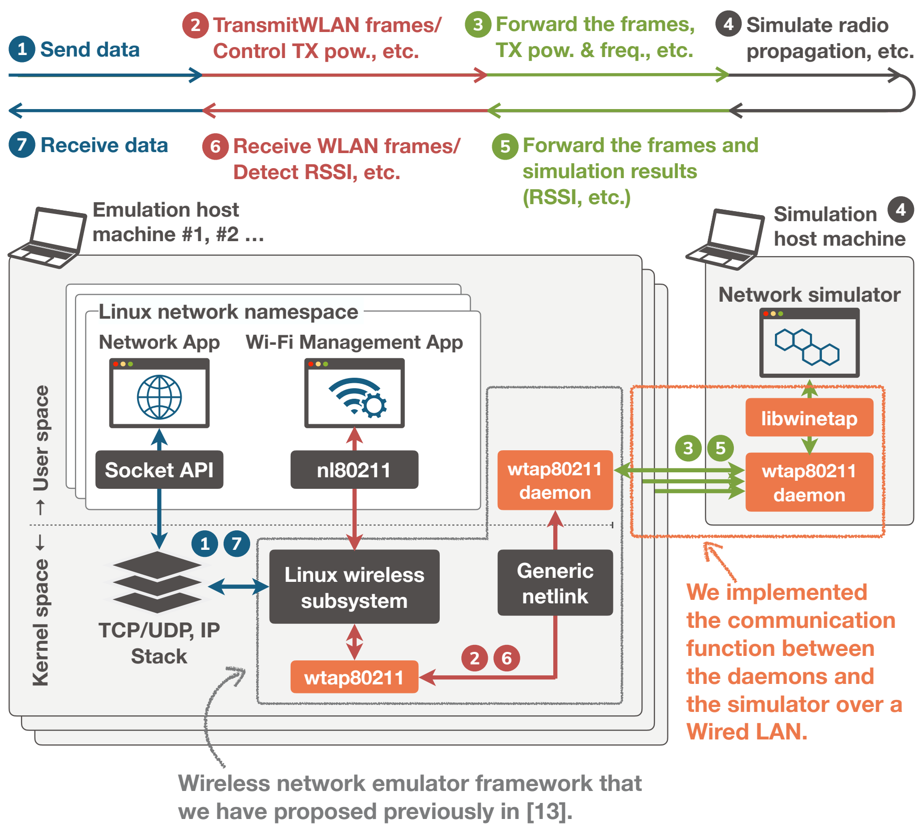
1 Send data
2 TransmitWLAN frames/ Control TX pow., etc.
3 Forward the frames, TX pow. & freq., etc.
4 Simulate radio propagation, etc.
7 Receive data
6 Receive WLAN frames/ Detect RSSI, etc.
5 Forward the frames and simulation results (RSSI, etc.)
Emulation host machine #1, #2 ...
Simulation host machine 4
Linux network namespace
Network App
Wi-Fi Management App
Socket API
nl80211
Network simulator
libwinetap
wtap80211 daemon
3 5
wtap80211 daemon
User space
Kernel space
1 7
TCP/UDP, IP Stack
Linux wireless subsystem
Generic netlink
wtap80211
2 6
We implemented the communication function between the daemons and the simulator over a Wired LAN.
Wireless network emulator framework that we have proposed previously in [13].

Figure 2.3: WiNE-Tap architecture

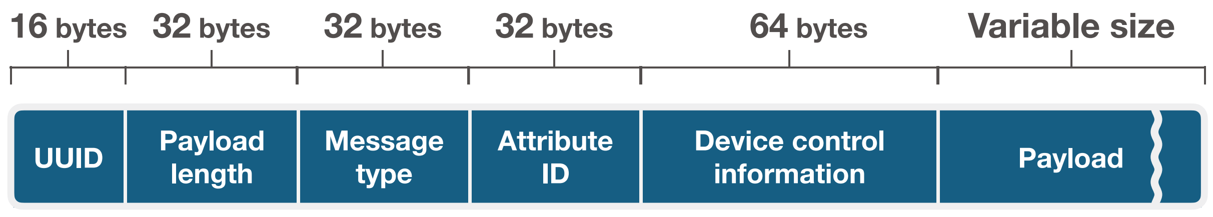

Figure 2.4: Message format used in `wtap80211`

lation host machines increase the transmission delays between `wtap80211` and the network simulator and make WiNE-Tap lose the real-time property of its emulation results. Therefore, in WiNE-Tap, `wtap80211-daemon` packs an IEEE 802.11 frame into a jumbo frame and transfer the jumbo frame to another `wtap80211-daemon`.

`wtap80211-daemon` provides APIs based on the Unix domain socket[92] for user applications to communicate with `wtap80211-daemon`. The Linux system supports a variety of inter-process communication mechanisms such as pipeline, filesystem, memory-mapped files, shared memory, and sockets. Memory-based inter-process communication systems are faster than other systems because of no access to secondary storage devices. However, memory-based inter-process communication requires mutual exclusion to avoid conflicting access to the memory and makes it harder to manage the mutual exclusion as the number of network nodes increases in the host machine. Meanwhile, an unix domain socket realizes lightweight inter-process communication and provides similar socket APIs to TCP/IP sockets, it makes it easy for the user applications to communicate with `wtap80211 daemon` using socket APIs.

`Libwinetap` is the static library to help developers implement the function to communicate with `wtap80211-daemon` via the Unix domain socket. `Libwinetap` also provides APIs to receive and send messages, including IEEE 802.11 frames or the configuration parameters of the IEEE 802.11 network device with `wtap80211-daemon`. Figure 2.4 shows the message format sent from `wtap80211`. The UUID can only be used for `wtap80211` to identify the messages. The payload length is basically equal to the length of an IEEE 802.11 frame. The message type is used to identify whether the message carries the IEEE 802.11 frames or the IEEE 802.11 network device configuration. The device control information includes some data structures defined in the Linux wireless subsystem.

The data structures in the device control information are the same as the data structures used in the Linux wireless subsystem, such as `struct ieee80211_conf`, `struct ieee80211_bss_info`, `struct ieee80211_rx_status`, and `struct ieee80211_tx_options` defined in the Linux kernel source codes [93], except some data fields of the data structure containing the pointer to the kernel resources. Which data structure is packed in the device control information field is packed field depends on what configuration the Linux wireless subsystem changes. For example, if the message from `wtap80211` carries an IEEE 802.11 data frame for transmission, the device control information field has the data structure as the same as `struct ieee80211_tx_options`, which defines transmission power and center frequency. If the message is a notification related to `RFkill`, the device control information field is structured with `struct ieee80211_conf`, containing the bit flags of hardware power status. `RFkill` provides user applications with APIs to turn on/off the RF signal transmission and the power supply of IEEE 802.11 network devices.

However, the device control information field does not have multiple data structures simultaneously. For example, a user turns the operation mode of an IEEE 802.11 device to the ad-hoc mode, and the IEEE 802.11 Basic Service Set (BSS) and the software switch of the IEEE 802.11 network device power status is changed at the same time. Then `wtap80211` sends two messages that each has `struct ieee80211_bss_info` and `struct ieee80211_conf` in their device control information field.

2.4 Operation validation and performance evaluation

This section describes the performance evaluation of WiNE-Tap. The performance evaluation aims to check that WiNE-Tap works correctly and clarify the conditions for improving the real-time property of the emulation results.

2.4.1 Setting up the experiment environment

For the performance evaluation of WiNE-Tap, the author built an experimental implementation of WiNE-Tap as shown in Figure 2.5, which shows the configuration of the experiment environment for the performance evaluation. The performance evaluation first aims to check that WiNE-Tap sufficiently functions to emulate IEEE 802.11p-based vehicular network systems. Therefore, the experiment environment assumes to emulate an IEEE 802.11a Ad-hoc network consisting of two wireless network nodes.

The experiment environment comprises three physical machines. Two of the physical machines work as emulation host machines, and the other machine works as a simulation host machine. The emulation host machines execute Linux systems with the wireless network emulation framework and network applications. Each of the emulation host machines performs as a wireless network node. The simulation host machine executes a network simulator, Scenargie, and `wtap80211-daemon`. The three physical machines connect over two Gigabit Ethernet networks, the machine control network and the data transfer network. The machine control network is used to monitor the physical machines and sends commands necessary to manage WiNE-Tap. The data transfer network carries the messages, including IEEE 802.11 frames, and the configuration parameters exchanged between `wtap80211` and the network simulator. The reason for using the two Ethernet networks is to avoid the observer effect and accurately measure the practical performance of WiNE-Tap. The physical machines have the exact specification as shown in Table 2.3.

Table 2.3: Host machine specification

Component	Specification
Linux distribution/Kernel	Ubuntu 16.04.6 LTS / Linux 4.4.170-generic
CPU	AMD Athlon PRO 200GE @ 3.2 GHz
Memory	DDR4-2666 4 GB
Storage	SSD 120 GB / HDD 1 TB
Wi-Fi chips	RTL8822BE IEEE 802.11a/b/g/n/ac wireless card
Wi-Fi Antenna	Omnidirectional antenna

The emulation host machines form an IEEE 802.11a ad-hoc network that imitates an IEEE 802.11p-based vehicular ad-hoc network and transfers bulk data using iPerf3 and the Constant Bit Rate (CBR) application scripted by the author in Python. The emulation host machines have a wireless LAN card made by RealTek, RTL8822BE. In the real environment, the emulation host machines are far away with a 1-meter distance in the room where no access points communicate over 5 GHz bands. The validation of the emulation results of WiNE-Tap is performed by comparing with the measurements in the real environment. The emulated network and the real network are formed by using the same configuration. The validation of the emulation results is performed by comparing the emulated results with the measurements in the real environment

2.4.2 Operation validation during IEEE 802.11 link establishment

First, the author confirms that WiNE-Tap emulates the IEEE 802.11 link establishment in the virtual IEEE 802.11 Ad-hoc network by monitoring the logs output by the `wtap80211-daemon`. Figure 2.6 and Figure 2.7 show the logs captured from the emulation host machines while emulating the IEEE 802.11 link establishment. The lines in Figure 2.6 show the configuration sequence from the beginning of creating an ad-hoc network to the end of completing the association with another wireless network node. The detailed descriptions of the lines are written here:

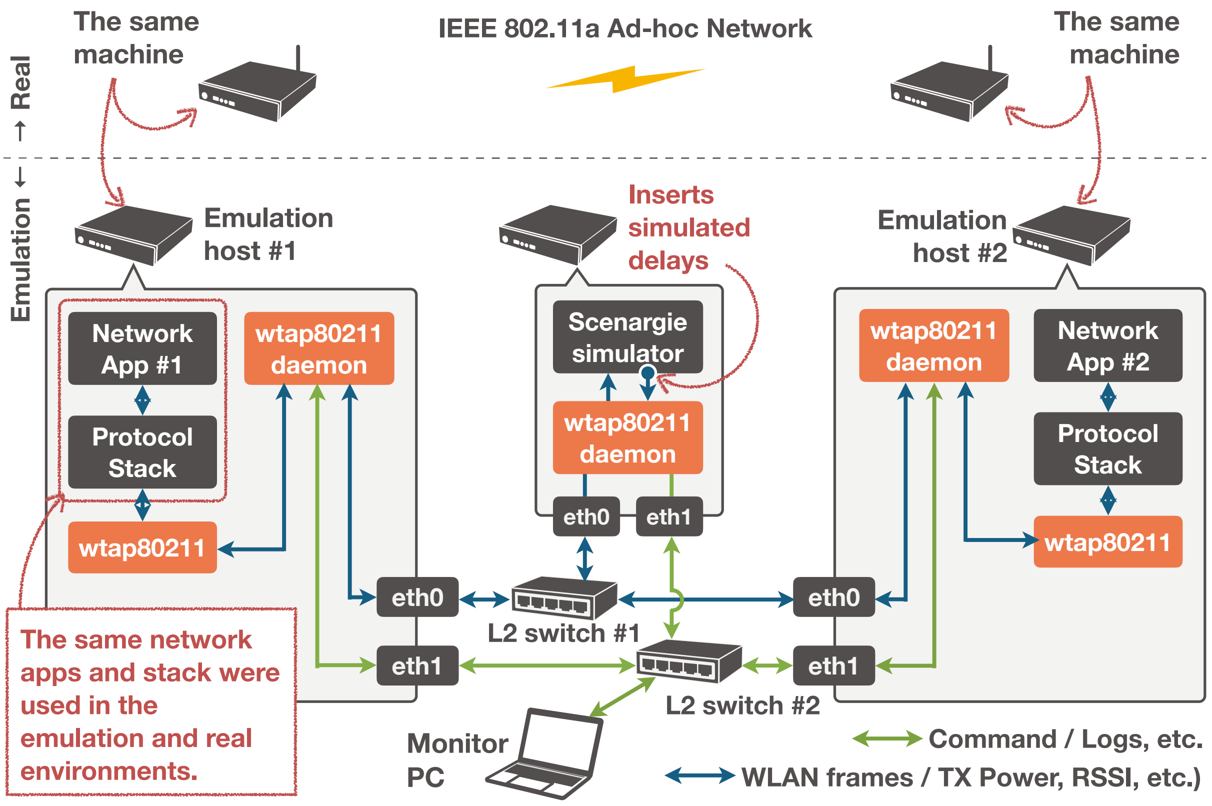

Figure 2.5: Configuration of the experiment environment for WiNE-Tap

- Line 1 to 6: The emulation host machine begins to create a new ad-hoc network and initialize the wireless network device whose hardware address is `0c:ff:fa:00:00:01`. Note that the virtual wireless network device emulated by `wtap80211` has a hardware address whose 24 bits from the most significant bit is `0c:ff:fa`. Thus, the wireless network device is `wtap80211`.

- Line 7 to 14: The new channel context information is created, and the transmission power and center frequency are set to the wireless network device. In this case, the transmission power and frequency are 20 dBm and 5180 MHz, respectively.

- Line 15 to 23: The contention window sizes of QoS queues are configured to the wireless network device. In this case, all the contention window sizes are the same because the QoS function is disabled.

- Line 25 to 37: The Basic Service Set (BSS) information is updated for the created ad-hoc network. Line 31 means the beacon interval is set to 102,400 μs = 102.4 ms. The value comes from the equation: $100 \times 1[\text{TU}] = 100 \times 1024[\mu\text{s}] = 102.4[\text{ms}]$. The Time Unit (TU) is a standard time unit used in IEEE 802.11 standards. Line 36 means that beaconing is enabled to disseminate IEEE 802.11 advertisement frames to notify other wireless network devices of the ad-hoc network. Line 38 means that the Service Set ID (SSID) of the ad-hoc network is set to `74:65:73:74:00:00`, which is `test` in ASCII code.

Figure 2.7 shows the logs captured from the other emulation host machine. The emulation host machine is the client of the ad-hoc network. The lines shown in Figure 2.7 are almost the same as the lines shown in Figure 2.6. The differences between Figure 2.6 and Figure 2.7 are written below:

- Line 1 to 10: the wireless network device whose hardware address is `0c:ff:fa:00:00:00` is turned on by a user. Concretely, the idle flag is true in Line 2, but it turns false in Line 7.

- Line 37 to 40: The wireless network device joined the ad-hoc network whose SSID is `74:65:73:74:00:00` (that is an ASCII string, “test.”), which is the same SSID of the ad-hoc network created by the emulation host machine #1.

```
1. [ 4627.491892] wlan1: Creating new IBSS network, BSSID 56:60:7e:70:76:12
2. [ 4627.491899] wtap80211d:  info: wtap_ops_config: HWconfig changed (HWaddr = 0c:ff:fa:00:00:01)
3. [ 4627.491902] wtap80211d:  info: wtap_ops_config:   freq = 0, idle = 0, ps = 0, smps = automatic
4. [ 4627.491904] wtap80211d:  info: wtap_ops_config:   ps_dtim_period = 0, ps_timeout = 0
5. [ 4627.491907] wtap80211d:  info: wtap_ops_config:   radar detection disabled
6. [ 4627.491909] wtap80211d:  info: wtap_ops_config:   minimum TX power changed 0 => 0 [dBm]
7. [ 4627.491923] wtap80211d:  info: wtap_ops_add_chanctx: channel context added (HWaddr = 0c:ff:fa:00:00:01, ctx_id =
   0xc969be6f): 5180[MHz/width], freq1: 5180[MHz], freq2: 0[MHz] max_power = 20[dBm]
8. [ 4627.491926] wtap80211d:  info: wtap_ops_assign_vif_chanctx: assigned vif (addr = 0c:ff:fa:00:00:01, id =
   0x536d3658) to channel context (freq = 5180[MHz], max_power = 20[dBm] ctx_id = 0xc969be6f)
9. [ 4627.491928] wtap80211d:  info: wtap_ops_assign_vif_chanctx: vif->chanctx_conf is null
10.[ 4627.491931] wtap80211d:  info: wtap_ops_bss_info_changed: BSS info changed (HWaddr: 0c:ff:fa:00:00:01):
11.[ 4627.491934] wtap80211d:  info: wtap_ops_bss_info_changed:   TX power changed: 20 => 20 [dBm]
12.[ 4627.491937] wtap80211d:  info: wtap_ops_bss_info_changed: BSS info changed (HWaddr: 0c:ff:fa:00:00:01):
13.[ 4627.491940] wtap80211d:  info: wtap_ops_bss_info_changed:   NIC status: active -> active
14.[ 4627.491944] wtap80211d:  info: wtap_ops_change_chanctx: channel context changed (HWaddr = 0c:ff:fa:00:00:01,
   ctx_id = 0xc969be6f): 5180[MHz/width], freq1: 5180[MHz], freq2: 0[MHz], max_power = 20[dBm]
15.[ 4627.491947] wtap80211d:  info: wtap_ops_conf_tx: TX queue parameters changed:
16.[ 4627.491950] wtap80211d:  info: wtap_ops_conf_tx:   queue = 0, txop = 0, cw_min = 15, cw_max = 1023, aifs = 2)
17.[ 4627.491953] wtap80211d:  info: wtap_ops_conf_tx: TX queue parameters changed:
18.[ 4627.491956] wtap80211d:  info: wtap_ops_conf_tx:   queue = 1, txop = 0, cw_min = 15, cw_max = 1023, aifs = 2)
19.[ 4627.491959] wtap80211d:  info: wtap_ops_conf_tx: TX queue parameters changed:
20.[ 4627.491961] wtap80211d:  info: wtap_ops_conf_tx:   queue = 2, txop = 0, cw_min = 15, cw_max = 1023, aifs = 2)
21.[ 4627.491964] wtap80211d:  info: wtap_ops_conf_tx: TX queue parameters changed:
22.[ 4627.491967] wtap80211d:  info: wtap_ops_conf_tx:   queue = 3, txop = 0, cw_min = 15, cw_max = 1023, aifs = 2)
23.[ 4627.491970] wtap80211d:  info: wtap_ops_bss_info_changed: BSS info changed (HWaddr: 0c:ff:fa:00:00:01):
24.[ 4627.491972] wtap80211d:  info: wtap_ops_bss_info_changed:   QoS: enabled => disabled
25.[ 4627.491975] wtap80211d:  info: wtap_ops_bss_info_changed: BSS info changed (HWaddr: 0c:ff:fa:00:00:01):
26.[ 4627.491977] wtap80211d:  info: wtap_ops_bss_info_changed:   ERP CTS Prot: 0 => 0
27.[ 4627.491979] wtap80211d:  info: wtap_ops_bss_info_changed:   ERP Preamble: 0 => 0
28.[ 4627.491982] wtap80211d:  info: wtap_ops_bss_info_changed:   ERP Slot: 0 => 0
29.[ 4627.491984] wtap80211d:  info: wtap_ops_bss_info_changed:   HT info: 0xb => 0xb
30.[ 4627.491986] wtap80211d:  info: wtap_ops_bss_info_changed:   Basic rate: 0 => 1
31.[ 4627.491988] wtap80211d:  info: wtap_ops_bss_info_changed:   Beacon interval: 0 -> 102400
32.[ 4627.491991] wtap80211d:  info: wtap_ops_bss_info_changed:   BSSID: 00:00:00:00:00:00 => 56:60:7e:70:76:12)
33.[ 4627.492037] wtap80211d:  info: wtap_ops_bss_info_changed:   Beacon info:
34.[ 4627.492040] wtap80211d:  info: wtap_ops_bss_info_changed:   Beaconing: disabled => enabled
35.[ 4627.492046] wtap80211d:  info: wtap_ops_bss_info_changed:   IBSS: joind, (new IBSS created)
36.[ 4627.492049] wtap80211d:  info: wtap_ops_bss_info_changed:   SSID: 00:00:00:00:00:00 => 74:65:73:74:00:00
   (unvisible)
37.[ 4627.492087] IPv6: ADDRCONF(NETDEV_CHANGE): wlan1: link becomes ready
```

Figure 2.6: Logs captured from the emulation host machine #1 during emulating the IEEE 802.11 link establishment

For these results, Figure 2.6 and Figure 2.7 show that the IEEE 802.11 link establishment is emulated in the virtual network on WiNE-Tap.

2.4.3 RTTs and throughput comparisons between the real and emulated network

Figure 2.8 shows the cumulative distribution function (CDF) of round trip times measured in the emulated network and the real network when the two network nodes generate UDP data traffic with 1.5 Mbps. The ideal emulated RTTs depicted as the red dashed curve in Figure 2.8 is the emulated results minus the mode for the emulated results, 0.61 ms, which is the transmission delay between the two emulation host machines in the Ethernet network. The results indicate the emulated RTTs minus the transmission overhead in the Ethernet network is almost the same as the real environment.

Figure 2.9 shows the results of UDP throughput measured in the emulated network and the real environment when the two network nodes generate UDP data traffic each other at the fixed data rate between 1.0 to 3.5 Mbps by the 0.5 Mbps interval. The emulated throughputs when the fixed bit rate is between 1.0 to 2 Mbps is the same as the real environments, but the throughputs when the fixed data rate is 2.5 Mbps or more degrades and is zero when the data rate is fixed at the 3.5 Mbps. When the data rate is 2.5 Mbps and 3.0 Mbps, 36 % and 60 % of UDP segments transmitted from the two network nodes are dropped by the network simulator, respectively. In addition, when the data rate is 3.5 Mbps, all the UDP segments from the two network nodes become unreachable.

2.4.4 Investigation of the performance degradation in the network simulator

The network simulator caused the performance degradation shown in the previous section with high possibility. There is another possibility that the wireless network emulation framework caused the performance degradation. However, it is confirmed in [94] that the wireless network emulation framework emulates the UDP data traffic with a data rate of more than 50 Mbps and has sufficient performance to emulate the IEEE 802.11 ad-hoc

```
1.[ 4627.589859] wtap80211d:  info: wtap_ops_config: HWconfig changed (HWaddr = 0c:ff:fa:00:00:00)
2.[ 4627.589861] wtap80211d:  info: wtap_ops_config:   freq = 0, idle = 1, ps = 0, smps = automatic
3.[ 4627.589863] wtap80211d:  info: wtap_ops_config:   ps_dtim_period = 0, ps_timeout = 0
4.[ 4627.589865] wtap80211d:  info: wtap_ops_config:   radar detection disabled
5.[ 4627.589868] wtap80211d:  info: wtap_ops_config:   minimum TX power changed 0 => 0 [dBm]
6.[ 4627.589875] wtap80211d:  info: wtap_ops_config: HWconfig changed (HWaddr = 0c:ff:fa:00:00:00)
7.[ 4627.589878] wtap80211d:  info: wtap_ops_config:   freq = 0, idle = 0, ps = 0, smps = automatic
8.[ 4627.589880] wtap80211d:  info: wtap_ops_config:   ps_dtim_period = 0, ps_timeout = 0
9.[ 4627.589883] wtap80211d:  info: wtap_ops_config:   radar detection disabled
10.[ 4627.589885] wtap80211d:  info: wtap_ops_config:   minimum TX power changed 0 => 0 [dBm]
11.[ 4627.589894] wtap80211d:  info: wtap_ops_add_chanctx: channel context added (HWaddr = 0c:ff:fa:00:00:00, ctx_id
  = 0x6151e63): 5180[MHz/width], freq1: 5180[MHz], freq2: 0[MHz] max_power = 20[dBm]
12.[ 4627.589897] wtap80211d:  info: wtap_ops_assign_vif_chanctx: assigned vif (addr = 0c:ff:fa:00:00:00, id =
  0xa70d2bdb) to channel context (freq = 5180[MHz], max_power = 20[dBm] ctx_id = 0x6151e63)
13.[ 4627.589899] wtap80211d:  info: wtap_ops_assign_vif_chanctx: vif->chanctx_conf is null
14.[ 4627.589902] wtap80211d:  info: wtap_ops_bss_info_changed: BSS info changed (HWaddr: 0c:ff:fa:00:00:00):
15.[ 4627.589904] wtap80211d:  info: wtap_ops_bss_info_changed:   NIC status: active -> active
16.[ 4627.589909] wtap80211d:  info: wtap_ops_change_chanctx: channel context changed (HWaddr = 0c:ff:fa:00:00:00,
  ctx_id = 0x6151e63): 5180[MHz/width], freq1: 5180[MHz], freq2: 0[MHz], max_power = 20[dBm]
17.[ 4627.589912] wtap80211d:  info: wtap_ops_conf_tx: TX queue parameters changed:
18.[ 4627.589915] wtap80211d:  info: wtap_ops_conf_tx:   queue = 0, txop = 0, cw_min = 15, cw_max = 1023, aifs = 2)
19.[ 4627.589918] wtap80211d:  info: wtap_ops_conf_tx: TX queue parameters changed:
20.[ 4627.589921] wtap80211d:  info: wtap_ops_conf_tx:   queue = 1, txop = 0, cw_min = 15, cw_max = 1023, aifs = 2)
21.[ 4627.589924] wtap80211d:  info: wtap_ops_conf_tx: TX queue parameters changed:
22.[ 4627.589926] wtap80211d:  info: wtap_ops_conf_tx:   queue = 2, txop = 0, cw_min = 15, cw_max = 1023, aifs = 2)
23.[ 4627.589930] wtap80211d:  info: wtap_ops_conf_tx: TX queue parameters changed:
24.[ 4627.589932] wtap80211d:  info: wtap_ops_conf_tx:   queue = 3, txop = 0, cw_min = 15, cw_max = 1023, aifs = 2)
25.[ 4627.589935] wtap80211d:  info: wtap_ops_bss_info_changed: BSS info changed (HWaddr: 0c:ff:fa:00:00:00):
26.[ 4627.589938] wtap80211d:  info: wtap_ops_bss_info_changed:   QoS: disabled => disabled
27.[ 4627.589941] wtap80211d:  info: wtap_ops_bss_info_changed: BSS info changed (HWaddr: 0c:ff:fa:00:00:00):
28.[ 4627.589943] wtap80211d:  info: wtap_ops_bss_info_changed:   ERP CTS Prot: 0 => 0
29.[ 4627.589945] wtap80211d:  info: wtap_ops_bss_info_changed:   ERP Preamble: 0 => 0
30.[ 4627.589948] wtap80211d:  info: wtap_ops_bss_info_changed:   ERP Slot: 0 => 0
31.[ 4627.589950] wtap80211d:  info: wtap_ops_bss_info_changed:   HT info: 0xb => 0xb
32.[ 4627.589952] wtap80211d:  info: wtap_ops_bss_info_changed:   Basic rate: 0 => 1
33.[ 4627.589954] wtap80211d:  info: wtap_ops_bss_info_changed:   Beacon interval: 0 -> 102400
34.[ 4627.589957] wtap80211d:  info: wtap_ops_bss_info_changed:   BSSID: 36:51:fa:c5:ff:88 => 56:60:7e:70:76:12)
35.[ 4627.589959] wtap80211d:  info: wtap_ops_bss_info_changed:   Beacon info:
36.[ 4627.589961] wtap80211d:  info: wtap_ops_bss_info_changed:   Beaconing: disabled => enabled
37.[ 4627.589963] wtap80211d:  info: wtap_ops_bss_info_changed:   IBSS: joind,
38.[ 4627.589966] wtap80211d:  info: wtap_ops_bss_info_changed:   SSID: 74:65:73:74:00:00 => 74:65:73:74:00:00
  (unvisible)
39.[ 4627.589989] wtap80211d:  info: wtap_ops_sta_add: sta added (sta_id = 0x5c816e97) (HWaddr = 0c:ff:fa:00:00:00)
40.[ 4627.691920] wtap80211d:  info: wtap_ops_sta_add: sta added (sta_id = 0x4bc801b2) (HWaddr = 0c:ff:fa:00:00:01)
```

Figure 2.7: Logs captured from the emulation host machine #2 during emulating the IEEE 802.11 link establishment

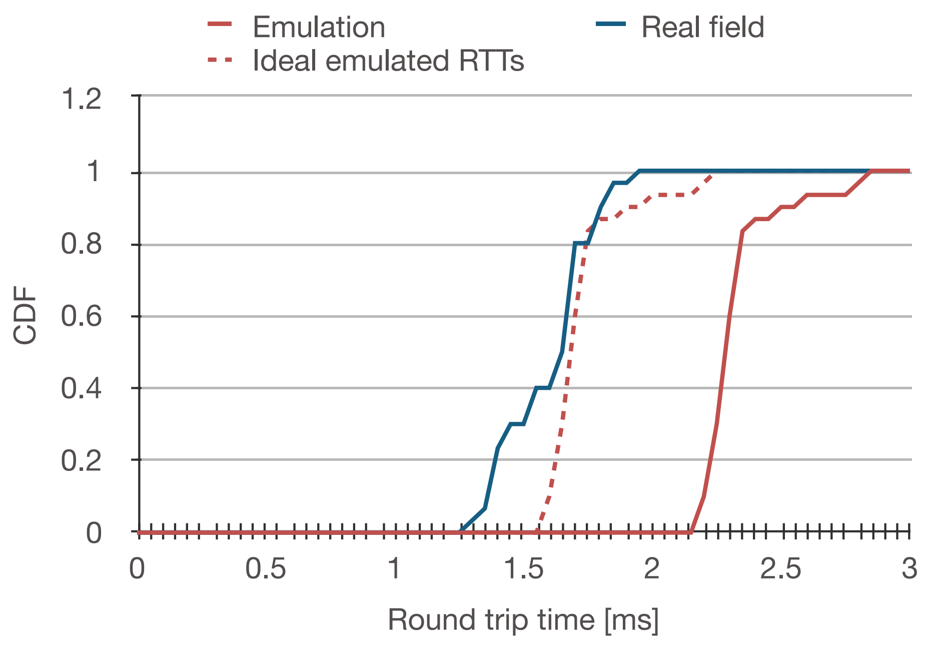

Figure 2.8: Round Trip Time (RTT) comparison between real and emulated environments

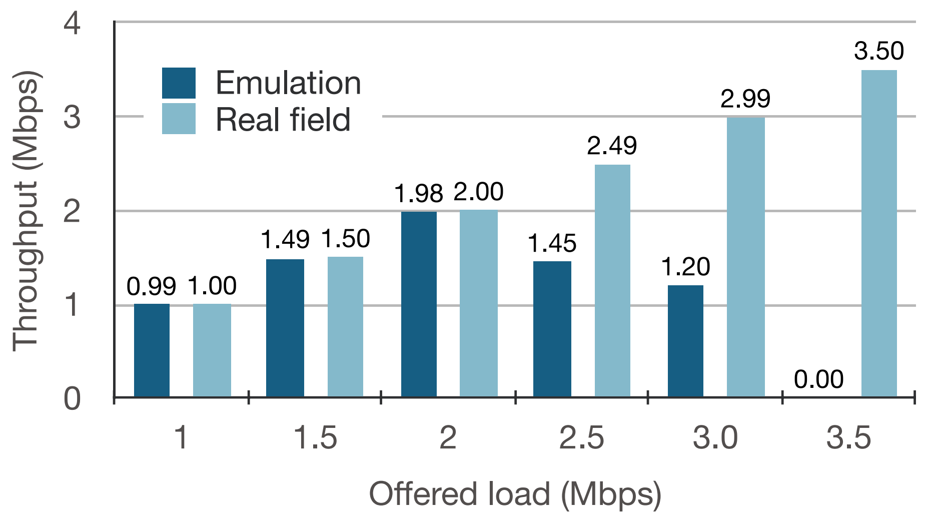

Figure 2.9: UDP throughput comparison between emulation and real environments

network discussed in this study.

Clarifying what degrades the UDP throughput in the network simulator, the author captures all the function calls while the two network nodes generate UDP data traffic with the data rate of 2.5 Mbps and depicts the hierarchy and the number of the function calls on the flame graphs[95].

Figure 2.10 is the flame graph of all the function calls captured while the two network nodes generate UDP data traffic at the data rate of 2.5 Mbps. A flame graph visualizes the hierarchy of function calls and the CPU usage ratio that each function occupies. The width and height of the flame graph show the CPU usage ratio and the depth of the call hierarchy, respectively. The flame graph is created from 986,592 function calls executed in the network emulation in 30 seconds. The flame graph shows the network simulator occupies a CPU usage ratio of 41.35%. The wireless network emulation framework occupies the CPU usage ratio of 27.93%. Other user processes occupy the remainder of the CPU ratio, 30.72%.

Figure 2.10 indicates the user processes related to WiNE-Tap operations, i.e., the network simulator and the wireless network emulation framework, totally occupy the CPU ratio of 69.28%. The CPU usage ratio of 70% is not low and certainly harms the machine's performance. However, it is difficult to think that the high CPU load critically influences the performance of the network simulator and makes the network simulator drop all UDP segments.

Figure 2.11 is the magnified flame graph of Figure 2.10, focussing the network simulator only. Figure 2.11 indicates that the system calls to obtain the current real-time such as `gettimeofday()` and `clock_gettime()` occupies almost all of the CPU usage ratio of the network simulator. The network simulator uses these system calls to check the timestamp recorded in the header of IEEE 802.11 frames. The network simulator validates the transmission order of IEEE 802.11 frames from the results of radio propagation and mobility models and obtains the current real-time to write the time to the logs.

2.4.5 Performance improvement by High Precision Event Timer

The high load to CPU resource caused by the system calls to obtain the current real-time can be reduced by using the hardware clock that responds quicker than other system

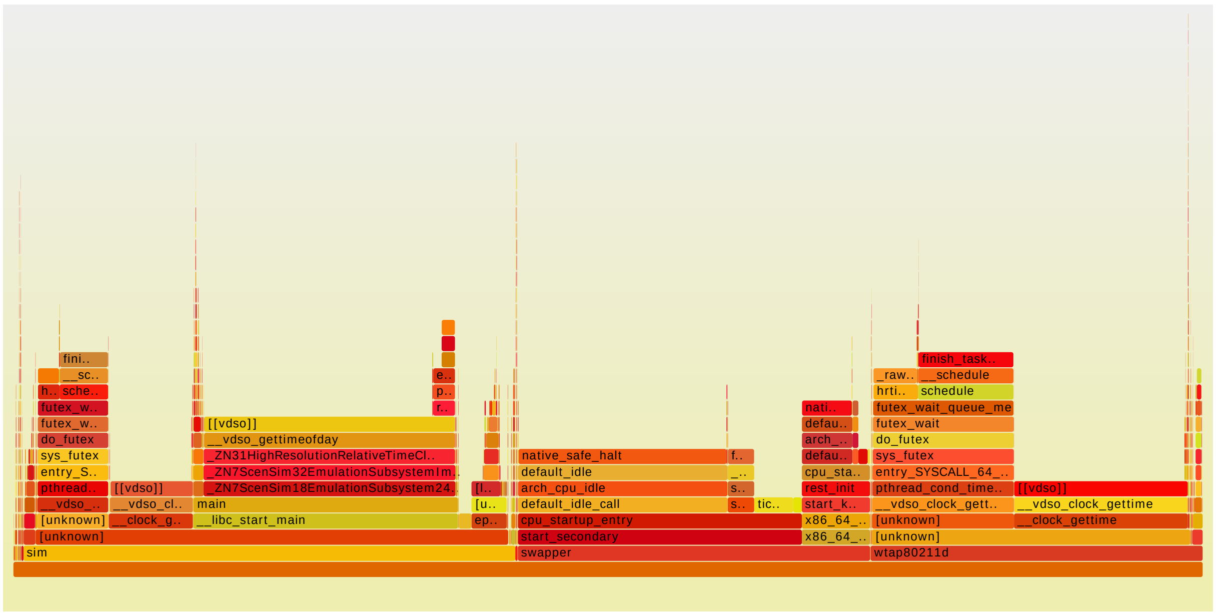

Figure 2.10: Flame graph of all the function calls during generating UDP data traffic with 2.5 Mbps

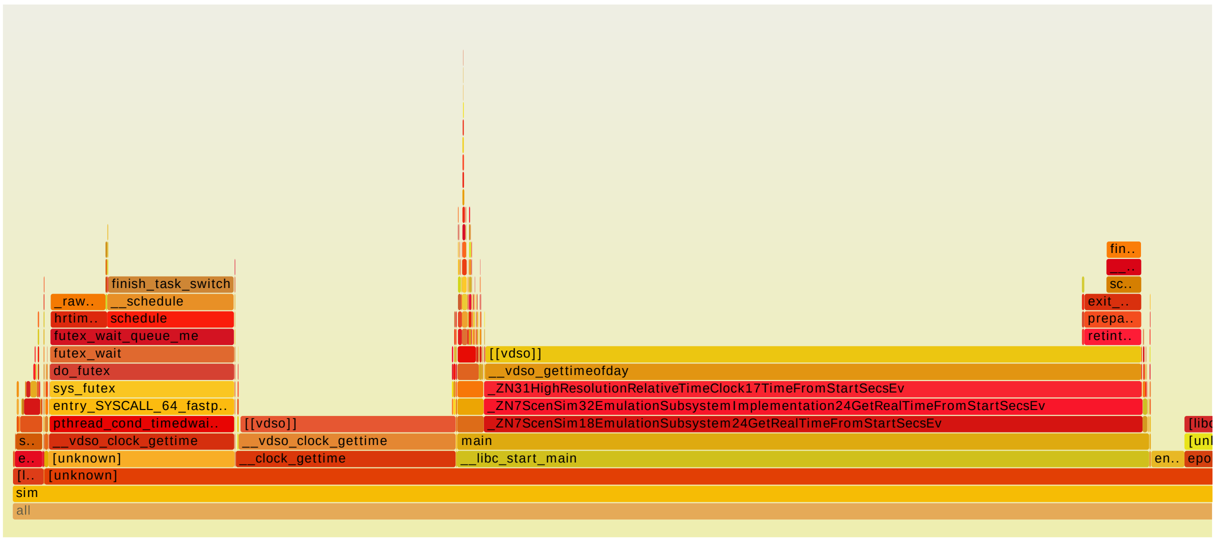

Figure 2.11: Flame graph of the function calls by the network simulator during generating UDP data traffic with 2.5 Mbps

clocks. x86-based machines have two kinds of hardware clocks. The one is ACPI PMT (Advanced Configuration and Power Interface Power Management Timer)[96] and HPET (High Precision Event Timer)[97]. Linux systems refer to ACPI PMT to obtain the current time due to the default kernel configuration, and the system clock in the Linux system is based on the ACPI PMT. The emulation results discussed in the previous section are measured with ACPI PMT.

ACPI PMT is less accurate and responds slower than HPET. The clock frequency of ACPI PMT is fixed at 3.58 MHz, but the clock frequency of HPET is at least 10 MHz or more. Therefore, HPET can improve the responsibility of the time clock system calls such as `gettimeofday()` and `clock_gettime()` and the performance of network emulation. For this reason, the author compares the emulation results measured with different hardware clocks.

Figure 2.12 and Figure 2.13 show the UDP throughput measured in the CBR application and iPerf3 when ACPI PMT is enabled. Since the measurements shown in Figure 2.9 may be influenced by the iPerf3's implementation, the author implemented the Constant Bit Rate (CBR) application in Python and compare the measurements obtained with iPerf3 and the CBR application. In addition, the measurement with iPerf3 and the CBR application is performed when using multiple machines and a single machine that executes both the WiNE-Tap and the network simulator or multiple machines. In the figures, the vertical bars in deep blue and light blue are the throughput measured in the real environment and the simulated network using Scenargie, respectively. The vertical bars in red and orange are the result of the emulation environment constructed with a single machine and multiple machines, respectively.

In Figure 2.13, when the configured data rate is 1 to 2 Mbps, the emulated data rate measured with iPerf3 is the same as the field and simulated performance. However, the data rate measured with iPerf3 degrades at 4 to 6 Mbps. Besides, the data rate is less than the field performance when the preconfigured data rate is 3.0 Mbps.

The CBR's performance with the multi-hosted emulation follows the field performance. Although iPerf3 tries to generate the bursty data traffic as much as possible, the CBR application generates data traffic at the constant data rate. In addition, the computing load is distributed to multiple machines. For this reason, the CBR's performance is better

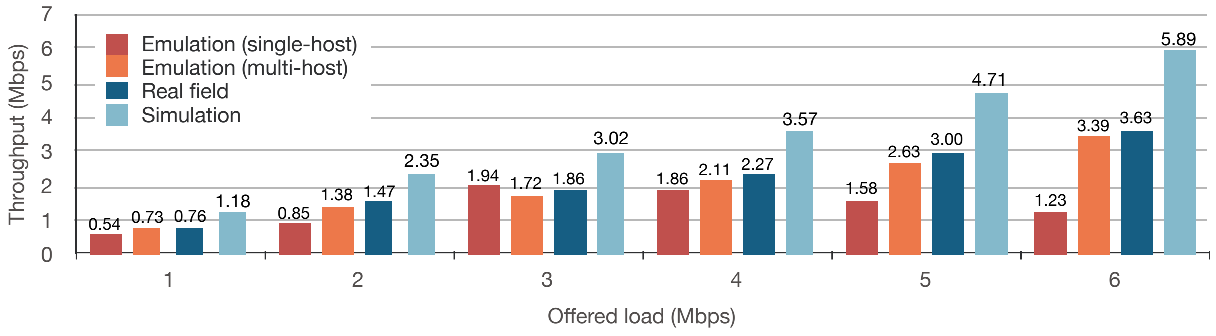

Figure 2.12: Throughput measured with CBR application and ACPI PMT

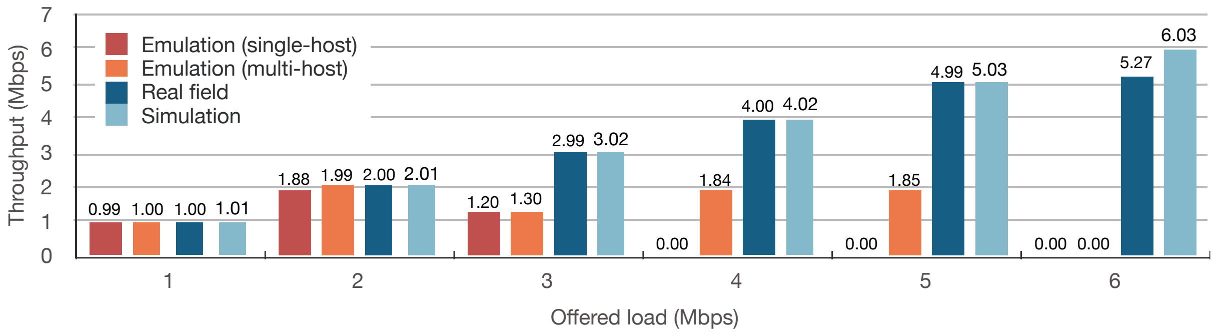

Figure 2.13: Throughput measured with iPerf3 and ACPI PMT

than other cases.

The performance degradation shown in the iPerf3's results was caused by two factors: bursty data traffic and Scenargie's implementation. In the network emulation, Scenargie must execute the simulation event such as frame transmission/reception and the move of network nodes in real-time and should be strict with the time when the events are fired.

Concretely, Scenargie enqueues the IEEE 802.11 frame from `wtap80211-daemon`and schedules the frame transmission event in the simulated network. When processing the frame transmission event, Scenargie calculates the propagation delay based on its radio propagation model and schedules the frame reception event in the simulated network based on the propagation delay. In other words, the frame reception event is fired after the simulated propagation delay gains from the frame transmission event. When processing the frame reception event, Scenargie validates that the current real-time does not pass the scheduled time that gains the propagation delay from the frame transmission time. However, Scenargie can drop the frame at the frame reception event when the current time and the time at which the frame reception event is fired is longer than the event synchronization interval, 1 second.

Since Scenargie calls the time clock, such as `gettimeofday()` and `clock_gettime()`, to check the time lags for every frame reception event, the frequent system calls apply the high computing load to the host machines. Especially when the emulation environment is constructed with a single machine, the significant high computing load is due to both the wireless network emulation framework and the network simulator concentrating the machine.

Figure 2.14 and Figure 2.15 are the CBR performance and iPerf3 performance when HPET is enabled. The performance results with HPET are better than ACPI PMT. Significantly, the CBR performance with HPET is close to the field performance. With HPET, the performance of the single-hosted network emulation degrades the performance of the multi-hosted one.

Figure 2.16 and Figure 2.17 show the CDF of RTTs measured in the real field and the single-hosted/multi-hosted network emulation environments. Figure 2.16 and Figure 2.16 are the results with ACPI PMT and HPET, respectively. The RTTs measured in the multi-hosted network emulation environment are almost the same as the field performance.

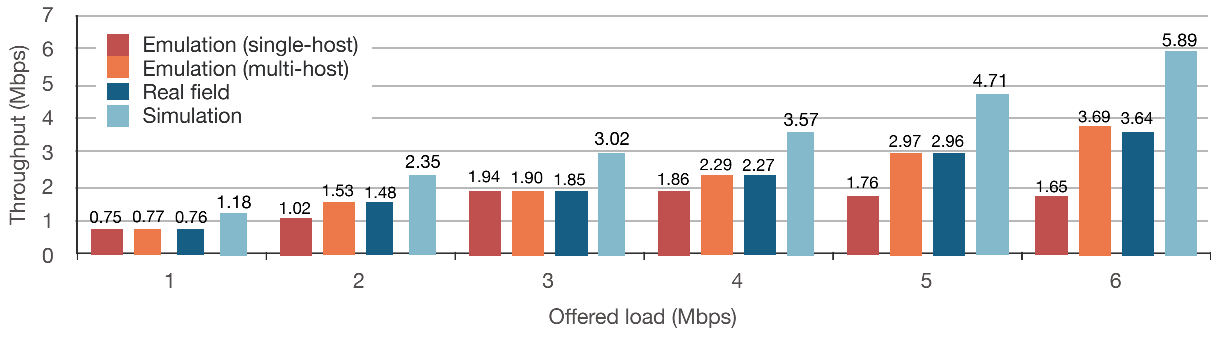

Figure 2.14: Throughput measured with CBR application and HPET

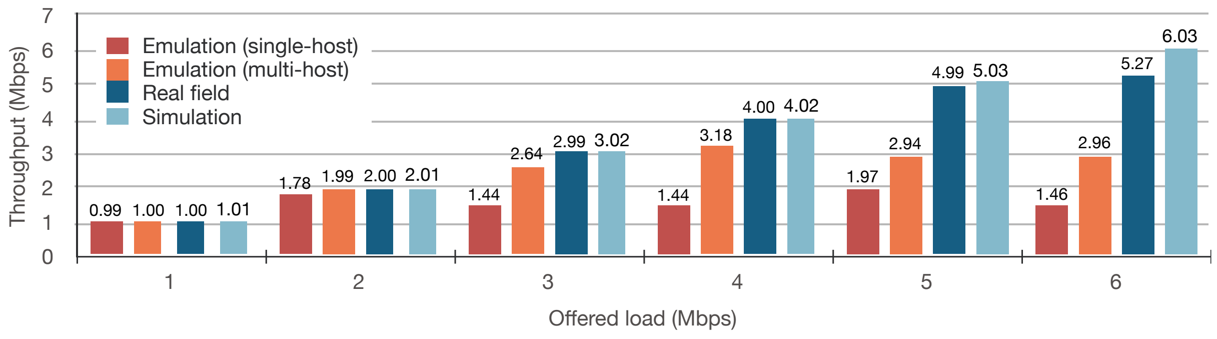

Figure 2.15: Throughput measured with iPerf3 and HPET

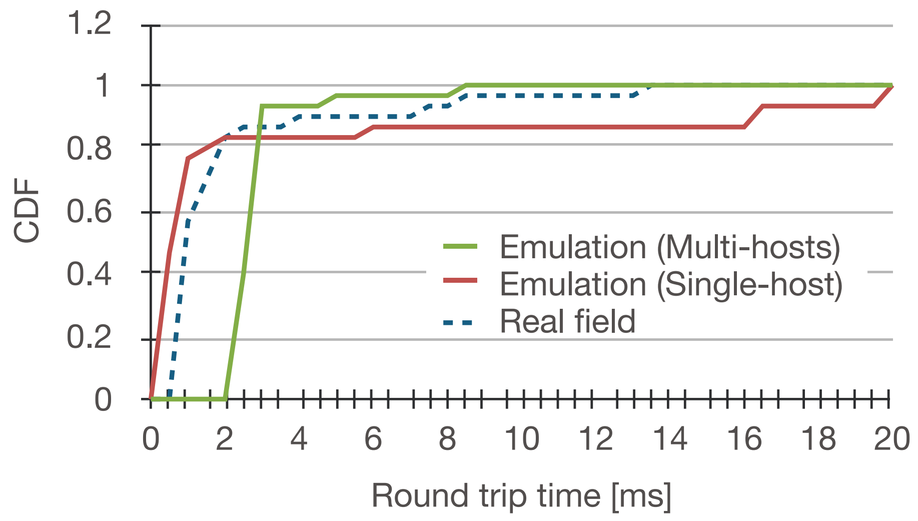

Figure 2.16: Round trip time measured with ACPI PMT

However, the RTTs measured performance with ACPI PMT is worse than the RTTs measured with HPET in the single-hosted network emulation environment. Since the performance comparison in Figure 2.12 to Figure 2.15 shows that HPET generates much more data traffic than ACPI, Scenargie processes many events in the single-/multi-hosted network emulation environment with HPET than APCI PMT. Therefore, RTTs measured in the single-hosted network emulation environment with HPET are longer than ACPI PMT.

Figure 2.18 shows the throughputs and RSSIs measured using the CBR application with the data rate of 4 Mbps when the two nodes are stationary with a 1-meter interval, and the one moves away from the other from the beginning of emulation. Figure 2.18 indicates that WiNE-Tap emulates throughputs that dynamically change following the change of RSSIs.

2.4.6 Discussion

Emulating the field performance as accurately as possible with the WiNE-Tap architecture can be improved by reducing the overhead of the time clock system calls such as `gettimeofday()` and `clock_gettime()`. Especially, replacing `gettimeofday()` with `clock_gettime()` improves the WiNE-Tap performance because the former consumes

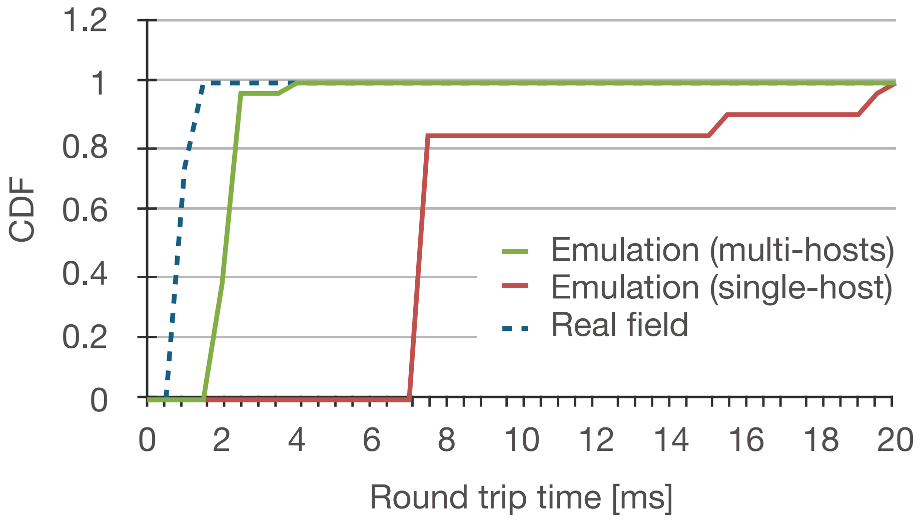

Figure 2.17: Round trip time measured with HPET

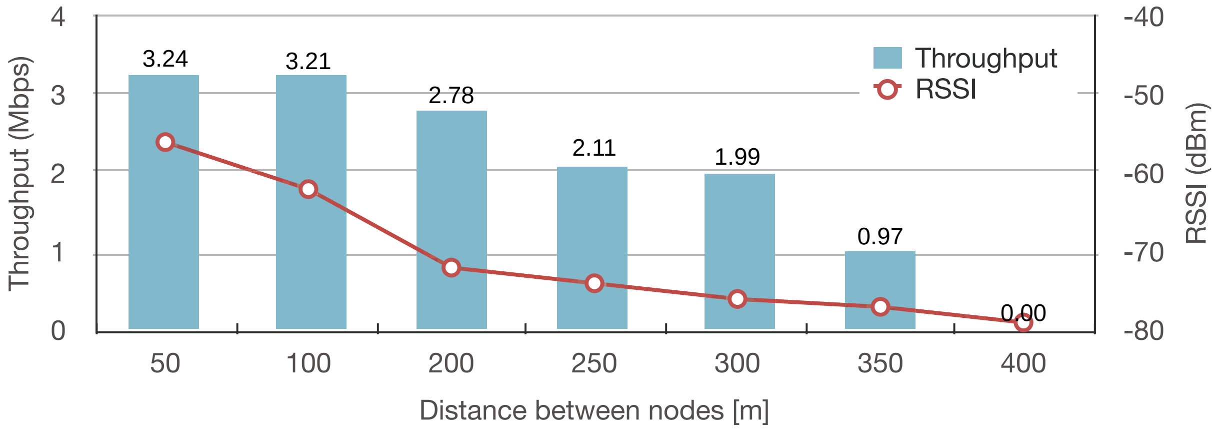

Figure 2.18: Change of throughput and received signal strength in the emulated network

more system resources than the latter. In addition, the monotonic clock, which responses to the monotonic time since an unspecified time, can improve the real-time property of the network simulator because it consumes small system resources. However, the monotonic clock requires the simulation and emulation host machines to synchronize their system clock and the basic time of the beginning of the monotonic time. If the monotonic clock without such time synchronization is used in the multi-hosted network emulation environment using WiNE-Tap, it becomes difficult to validate the correctness of the emulation results due to the error of timestamps between the machines.

The multi-hosted network emulation environment such as WiNE-Tap requires adjusting the propagation delay inserted during link emulation, considering the transmission delay over Ethernet. As shown in Figure 2.8, the RTTs in the emulated network contains the transmission delay over the Ethernet and are slightly longer than the field performance. The link emulation in the single-hosted network emulation environment does not require eliminating the transmission delay over Ethernet because data packets are captured locally, but the transmission delay over Ethernet is not ignorable for the multi-hosted network emulation environment. The network simulator should adjust and insert the simulated propagation delay based on the Ethernet overhead to eliminate the transmission delay over Ethernet. However, implementing the adjustment function on inserting delays is challenging because it requires modifying the existing radio propagation models that are confirmed to be correct.

Since the emulation results discussed in this study are measured using low-performance desktop PCs and restrict the network simulator's real-time property, the high-performance machines certainly enhance the performance of WiNE-Tap. It benefits the scalability of WiNE-Tap. As discussed above, the limited system resources restrict the network simulator's real-time property. In addition, other network simulators that consume fewer system calls to obtain the current time can help the performance improvement.

2.5 Conclusions

This chapter presented the novel wireless LAN emulator with wireless network tap devices (WiNE-Tap). WiNE-Tap enables the Linux IEEE 802.11 protocol stacks to operate in the virtual network and work interactively with radio propagation and mobility simulation behavior by a network simulator. The performance evaluation shows that WiNE-Tap has the performance meeting the requirement of vehicular network system development.

Chapter 3 Link Setup Time and bulk data transmission investigation in V2V

This chapter describes the first empirical investigation of initial link setup time and bulk data transmission in IEEE 802.11-based inter-vehicular communications. The study shows the link setup time reduction and the improvement of bulk data transmission size by Fast Initial Link Setup (FILS) in inter-vehicular communications based on real field measurements with comparison of Wi-Fi Protected Access version 2 Protected EAP protocol (WPA2-PEAP). This chapter is written based on the author's paper [14].

3.1 Introduction

This chapter introduces a vehicular network system designed to be used as a communication system under disaster conditions, as a concrete example of vehicular network systems that require inter-vehicular authentication and bulk data transmission. The vehicular network system is expected to contribute disaster response and is one of the vehicular network systems most expected to be realized. This section shows the problems on developing the vehicular network systems, following the problems of the current Japanese disaster communication systems and the introduction of vehicular disaster communication system introduction, and explains the motivation for conducting the first empirical investigation of inter-vehicular link setup time reduction and the increase of inter-vehicular bulk transmission data size by FILS.

3.1.1 Problems of the current disaster communication systems in Japan

Like the saying, ‘’One picture is worth a thousand words,” visual information such as photos and videos gives people more straightforward, intuitive information and helps them conduct rescue activity and evacuation as quickly as possible. On the other hand, as of 2021, the major means of communication under disaster conditions in Japan is based on voice messaging by the disaster prevention radio communication systems[98]. The disaster prevention radio communication systems are categorized into the broadcasting system and the locomotory system. The broadcasting system consists of heterogeneous broadcasting networks such as radio communication networks operated by municipalities, FM radio networks, and cable TV networks and disseminate voice messages such as early earthquake/tsunami alerts to residents through public address wireless loudspeakers or radio receivers, which are dispersively installed in a city and utilize 60 MHz-band radio. The locomotory system consists of base stations built on municipal buildings and mobile communication devices such as transceivers and car-mounted radio equipment. The locomotory system utilizes 260 MHz bands and provides voice/digital communication services. However, disaster prevention radio communication systems have several problems.

- Disaster prevention radio communication systems do not provide residents with a means of communication with disaster management agencies. Although the locomotory system provides users with bidirectional communication, it is not open to residents. For this reason, the residents cannot inform their dangers and rescue requests via disaster prevention radio communication systems.
- The voice messages cannot inform visual information such as photos and videos. Since the voice messages have less information than visual information and are recorded in Japanese, the residents and foreign people cannot possibly understand the impending dangers.

- The voice messages hardly reach the residents who have auditory disturbance or are far away from the speakers or under heavy rains.
- The maximum bitrate is 4.8 kbps with 4-level frequency shift keying (FSK), which is insufficient to transfer visual information such as photos and videos.

It is difficult to inform people of what people see in disaster-stricken areas by their voices because voice messages include no visual information and contain more or less vague information. The talker should inform listeners of what he/she sees correctly if he/she meets the disasters, but the talkers have their impressions and express what they see with different words based on their impressions. For example, Figure 3.1 is a photo capturing a landslip in Heavy rain of July, 2018. The photo makes it easy to understand how the landslip damages that place intuitively. However, it is difficult to explain what the picture captures concretely with words. In fact, the Global Facility for Disaster Reduction and Recovery (GFDRR) recommends in their report[98] that they encourage the utilization of visual information in disaster conditions. Moreover, the GFDRR report comments that visual information is helpful to remove the comprehension and language barrier. Foreign people may not understand voice messages. Therefore, it is crucial to share visual information in disaster conditions.

A means of communication reliable and robust to the damage from natural disasters is the key to sharing visual information under disaster conditions because the demand for communication between the disaster control headquarters and disaster-stricken areas are highest in the dozens of hours after a disaster strikes the areas but the standing communication infrastructures, such as cellular and optical fiber networks, are vulnerable to blackout and installed equipment failure[98]. The window period with no means of communication in disaster conditions could harm the health and lives of humans[99] Accordingly, the demand for disaster communication is highest in the hyperacute phase hours when the standing communication infrastructures are damaged[98]. It is essential to implement a reliable, robust disaster communication system. The hyperacute phase means the duration of several days after a natural disaster occurs and in which rescue activities are the most active.

Public and private organizations have developed other disaster communication systems to realize a disaster communication system robust to natural disasters. These disaster

Figure 3.1: Photo capturing the damage of landslips in Heavy rain of July, Heisei 30 from Disaster Photo Database of Institute for Fire Safety and Disaster Preparedness (http://www.saigaichousa-db-isad.jp/)

communication systems partly cover the disadvantages of disaster prevention radio communication systems. Here are the introductions of the disaster communication systems to focus on the unsolved problems of the current disaster communication systems.

- **NerveNet** is a wireless mesh network system consisting of multiple wireless stations dispersively installed in a town[100]. The wireless stations form an IEEE 802.11n-based wireless mesh network. The wireless stations have enough batteries to work 72 hours in the immediate aftermath of a natural disaster. If the wireless stations are pre-installed in disaster management agencies and shelters before a natural disaster, they can provide inter-site communication. In contrast, although the wireless stations can be moved by a carrier, they do not provide inter-site communication service in areas without them.
- **Relay-by-Smartphone** is a Bluetooth-based device-to-device communication system consisting of smartphones used by evacuating residents[101]. Relay-by-Smartphone is designed to deliver evacuation routes to shelters and other disaster prevention information to residents during evacuation and does not require building pre-installed communication infrastructures like NerveNet. Like forming a bucket brigade, smartphones of evacuating residents exchange disaster prevention information by device-to-device communication when they pass by each other. In contrast, its connectivity depends on the density of smartphones in areas and is presumably lost after evacuation completes. The architecture of Relay-by-Smartphone does not guarantee that all residents receive the evacuation routes evenly.
- **Q-ANPI** is a safety confirmation system provided by Quasi-Zenith Satellite System (QZSS)[102]. QZSS is a Japanese satellite system and provides centimeter-accuracy positioning and satellite communication service. Q-ANPI gathers the status information of shelters, such as the number of evacuees and requisite materials, and informs disaster management agencies like municipalities of the status information over QZSS satellite communication. Residents can also refer to the status information via the Q-ANPI website[103]. QZSS can be hardly damaged by natural disasters such as earthquakes and tsunami, but it does not provide point-to-point communication services and requires special receivers to use the service directly.

These disaster communication systems are designed for specific use cases. NerveNet provides inter-site communication among disaster management agencies and shelters. Relay-by-Smartphone disseminates alerts and route information necessary for evacuation to residents. Q-ANPI provides the general condition of disaster management for local governments. While improving disaster management requires cooperation with governments and residents, the current disaster communication systems insufficiently consider disaster management that is collaborate with residents. In particular, the existing disaster communication systems depends on standing communication infrastructures such as fixed stations, cellular network, and satellite communication systems. These communication infrastructures require the pre-installation of network equipment at specific locations and lack the flexibility of constructing a network in any place. In contrast, the vehicular network can be constructed anytime, anywhere in disaster conditions, and is compatible with the needs for disaster response.

3.1.2 Disaster communication system based on VDTNs with heterogeneous wireless systems

Figure 3.2 shows the overview of the disaster communication system based on VDTNs with heterogeneous wireless communications (heterogeneous VDTNs). The disaster communication system with heterogeneous VDTNs is used among disaster control headquarters, shelters, and emergency vehicles equipped with onboard units. The disaster control headquarters, shelters, and emergency vehicles each have direct narrow-band wireless communication links such as LoRa and Digital Convenience Radio (DCR). DCR is standardized in ARIB STD T-98[104] and uses the 150, 351, or 467 MHz bands. The DCR bitrate depends on the modulation; for example, it is 4.8 kbps when using $\pi/4$ shift frequency shift keying (FSK) frequency-division multiple access (FDMA) communications. The narrow-band wireless communication has several kilometers of communication range and is used for transmitting small data such as text messages. For example, it broadcasts emergency messages such as tsunami/earthquake warnings and location notifications.

The emergency vehicles work as data carriers in the disaster communication system. They carry a large amount of data that cannot be transferred over the narrow-band wireless communication links, such as photos and videos. The emergency vehicles have

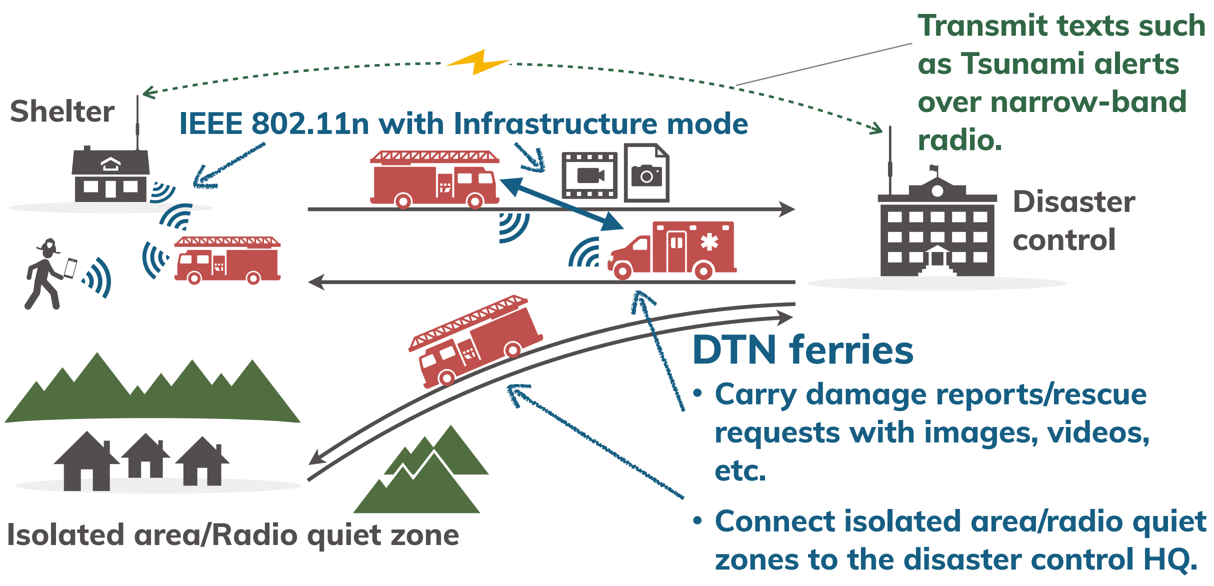
Transmit texts such as Tsunami alerts over narrow-band radio.
Shelter
IEEE 802.11n with Infrastructure mode
Disaster control
DTN ferries
• Carry damage reports/rescue requests with images, videos, etc.
• Connect isolated area/radio quiet zones to the disaster control HQ.
Isolated area/Radio quiet zone

Figure 3.2: Overview of the disaster communication system based on VDTNs with heterogeneous wireless communication: The figures depicts a situation where disaster management agencies such as fire departments use the system to establish communication links between the control center and remote places in the radio quiet zone.

onboard units and communicate over IEEE 802.11n protected by the enterprise authentication mechanisms such as WPA-PEAP.

In vehicular networks, vehicles generally form an ad-hoc network and communicate without link establishment procedures like the infrastructure mode. However, the disaster communication system allows the onboard units of the emergency vehicles to work in the infrastructure mode. They provide access point service and work as stations simultaneously and communicate with each other using the IEEE 802.11 link establishment procedure in the infrastructure mode. There are two reasons to use the infrastructure mode instead of the ad-hoc mode for inter-vehicular communication in the disaster communication system.

The onboard units can connect to personal mobile communication devices over IEEE 802.11n, such as smartphones and tablet PCs used by the members of the disaster management agencies. The members take photos and videos in disaster-stricken areas and report the damage of the disaster-stricken areas with the photos and videos. However, they can lose a means of communication to the disaster management headquarters in the disaster-stricken areas due to the failure of standing communication infrastructures such as cellular networks. In that case, they store the photos and videos to the storage of onboard units over Wi-Fi and request the emergency vehicles to carry data to the disaster management agencies.

The emergency vehicles must be protected by the enterprise authentication system managed by the disaster control headquarters to prevent the system from circulating misinformation. In the disaster communication system, the disaster management agencies exchange private and personal information necessary for rescue and medical activities, such as medical records and photos capturing disasters.

3.2 Problems and requirements on developing the VDTN-based disaster communication system

Security is one of the most important factor on developing the vehicular network system to protect the system from the malicious attacks and improve its reliability. Particularly, fast, secure link establishment is recommended to improve connectivity between highly mobile devices during inter-vehicular communications in vehicular delay/disruption tolerant networks (VDTN) used in the critical situation such as disaster-stricken areas[30]. Short link setup times reduce link establishment overhead and extend data transmission time. Secure links protect the vehicular network systems from tapping by unauthenticated users and misinformation circulation by malicious persons.

Individual authentication is necessary to protect against unauthorized tapping and prevent the spread of misinformation in disaster networks because they often carry sensitive information such as personal medical records. For example, the Wi-Fi Protected Access 2 Extensible Authentication Protocol (WPA2-EAP), which is also known as WPA2-Enterprise, is typically used for individual authentication on IEEE 802.11 systems, and it is also available in IEEE 802.11-based inter-vehicular communications. However, WPA2-EAP communications require a few seconds for link establishment because they must build Transport Layer Security (TLS) tunnels and exchange authentication information such as certificates and passwords. These long link establishment times reduce the time available for application data transferred between passing vehicles and thus the amount of data that can be transferred.

To minimize this problem, the newly proposed system uses the IEEE 802.11ai protocol, which is also known as the Fast Initial Link Setup (FILS), to reduce link establishment time and increase the transfer size of application data in the VDTN-based disaster networks. In operation, FILS enables access points and station nodes to authenticate each other within about 100 ms by using cached authentication information. The link setup time is defined as a period between when a station sends an IEEE 802.11 probe request to an access point and when the station obtains its own IP address from a DHCP server. On the other hand, inter-vehicular communication environment has unidentified factors that can affect the field performance of FILS, such as vehicles' mobility and radio propagation, and measuring the FILS performance in real vehicular environment is crucially important to identify the FILS practical performance.

To clarify the effect of link setup time reduction by FILS in intermittent inter-vehicular communications, this chapter describes laboratory and field experiments performed by the author to measure the FILS performance and reports the results of field experiments in this chapter. The field measurement results show that FILS reduces link setup time in intermittent inter-vehicular communications using 2.4 GHz IEEE 802.11n, which effectively increases the size of application data transmitted between passing vehicles by 10 MB compared with WPA2-PEAP. The IEEE 802.11ai was issued in 2017, and a few papers [105, 106, 107] have reported the FILS performance with mathematical models or network simulation. The next section will review related work on FILS performance evaluation in Section IV-E. However, to the best of my knowledge, no report to date has verified link setup time reduction via FILS in actual use, especially in vehicular networks. Therefore, the chapter represents the first report of a performance evaluation of FILS in a real-world vehicular DTN.

3.3 Related work

This section describes work related to individual authentication methods for IEEE 802.11 and vehicular networks and highlights the problems that may arise if the existing individual authentication methods are applied to IEEE 802.11-based inter-vehicular communications.

3.3.1 Individual authentication on IEEE 802.11

As stated above, the primary IEEE 802.11 individual authentication protocols are standardized in IEEE 802.11i/IEEE 802.1X[108, 109], which are also known as WPA2-EAP. These individual authentication protocols enable access points and stations to identify each other via certificates or username and passphrase pairs. EAP[110] has a variety of authentication/certification methods, such as EAP-TLS[111] and protected EAP (EAP-PEAP)[112]. EAP-TLS provides access points and stations with a method to identify each other via their certificates, but its certificate management process is complicated because it must distribute certificates to all stations. In contrast, EAP-PEAP allows access points to identify stations with their usernames and passphrases.

Unfortunately, even though EAP-PEAP simplifies certificate management, it requires an EAP exchange between an access point and station when setting up a secure link, as shown in Figure 3.3. In this process, the access point and the station need to exchange at least 22 frames between the beginning of the link setup and the completion of the station's IP address assignment. Additionally, the process can take a long time in situations where the frame loss rate is high, such as inter-vehicular communications, which reduces the time available to exchange data frames after the link setup, and thus the amount of data transmitted.

Xu et al. showed that the WPA2-EAP authentication delays increase as the number of vehicles rises in [113], which indicates that the WPA2-PEAP and client IP address assignment delays increase when the vehicle density is high and that throughput between the vehicles declines due to those delays.

Separately, other researchers have proposed methods to reduce the WPA2-EAP's latency between mobile devices with high mobility by pre-authentication or key caching. For example, Mishra et al.[114] proposed a proactive key distribution scheme using a neighbor graph, which shows the access points a station might possibly access after handoff. Their scheme allows a station to conduct authentication with access points to which the station may connect in the near future via the access point associated with the station. In an-

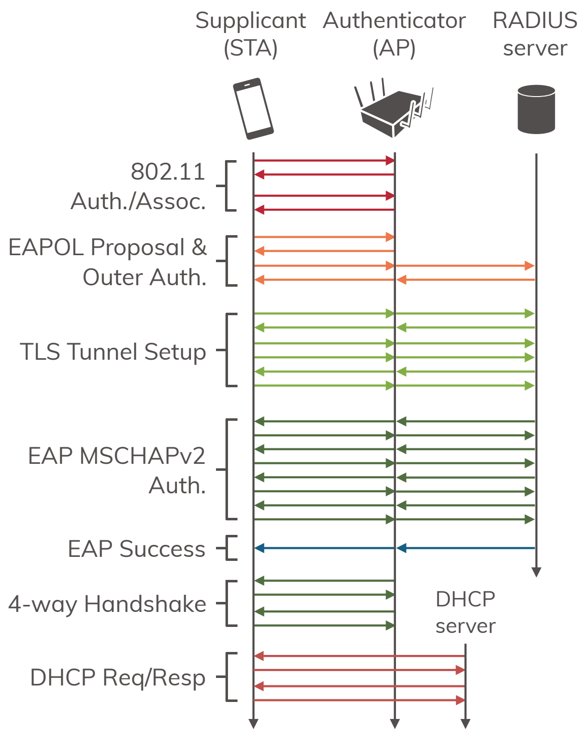

Figure 3.3: WPA2-PEAP frame sequences

other example, Hur et al. proposed a pre-authentication method for IEEE 802.11-based vehicular networks[115] that enhances Mishara's scheme and allows a station and an access point to cache a pairwise master key, which can reduce the authentication latency to a level lower than Mishara's method.

However, these schemes are inadequate for IEEE 802.11- based inter-vehicular communications because they assume that the access points are stationary and the station and access points are reachable. Those assumptions do not hold in IEEE 802.11-based inter-vehicular communications in disaster communication systems because emergency vehicles serving as DTN node ferries cannot rely on having sustain- able links when communication infrastructures fail. Further- more, emergency vehicles cannot link to other network nodes when they are in a radio blind zone.

3.3.2 Individual authentication on vehicular networks

Individual authentication protocols for inter-vehicular communications are standardized in IEEE 1609.2[116] and ETSI TS 103 097[117], which also define certificate-based authentication methods for vehicular networks. Certificate-based authentication methods help reduce authentication latency between vehicles because those vehicles only need to ex- change a few frames to complete the authentication process. This is simpler than username-passphrase-based authentication methods such as WPA2-PEAP.

Individual authentication protocols also require vehicles to have multiple certificates and acquire different certificates for every authentication because it is difficult to update vehicle certificates when there are no stable links between the vehicles and trust anchors. Therefore, the standards require the trust anchors (i.e., certificate authorities) to be responsible for vehicle certificate management, and thus bear the burden of managing numerous certificates. As a result, numerous researchers have looked for ways to facilitate certificate management on vehicular networks.

For example, Sun et al.[118] proposed a certificate update method using roadside units (RSUs), but that method depends on communication infrastructures, which makes it unsuitable for disaster communication systems because it is highly probable that RSUs will be unavailable in emergency situations. Separately, Feiri et al.[119] proposed a vehicle-based certificate distribution method that forces vehicles in close proximity to each other to proactively exchange certificates. However, Feiri's method would not work in low-vehicle-density areas where vehicles rarely encounter each other because the need to encounter other vehicles to keep certificates updated would degrade disaster communication systems. This is particularly true in systems where emergency vehicles can be sent to remote areas, such as mountainous regions, where they would have no chance to communicate with other emergency vehicles for extended periods.

Bohm et al.[120] proposed an IEEE 802.11p MAC enhancement that enables a roadside unit to share the authentication information of a car that has already been authenticated by another roadside unit. Since roadside units that have received the authentication information can authenticate the car with the information, they omit the authentication procedure with the car. Therefore, the MAC enhancement can reduce authentication overhead between roadside units and cars. However, the MAC enhancement does not support conventional security protocols such as WPA2-EAP and IEEE 1609.2 and only works with IEEE 802.11p.

Since, as indicated by the examples above, certificate-based authentication methods require stable links between emergency vehicles and trust anchors, as well as complicated certificate management procedures, they do not provide a realistic way to manage disaster communication systems. Therefore, the disaster communication system adopts the WPA2-PEAP username-passphrase-based authentication. Mano et al. reported the overhead of initial authentication on IEEE 802.11 harms seamless handover between high-mobility devices based on field experiment results in [121]. Since the abovementioned authentication and IP address assignment latency will presumably occupy communication time between moving vehicles, This study proposes using FILS for IEEE 802.11-based inter-vehicular communications to reduce individual authentication and IP address assignment delays.

3.4 Fast Initial Link Setup

This section describes the Fast Initial Link Setup (FILS) mechanisms standardized in IEEE 802.11ai, which enables the establishment of a secure connection within about 100,ms using the following four mechanisms:

- Channel scanning enhancement
- Active scanning optimization
- IP address assignment during the IEEE 802.11 associations
- Authentication information caching

3.4.1 Channel scanning enhancement

While authenticators broadcast a beacon frame every 100,ms, FILS also allows them to broadcast FILS discovery frames. Furthermore, while IEEE 802.11ai states that a FILS discovery frame must always have a Basic Service Set Identifier (BSSID) users have the option of including other information elements. Thus, supplicants can detect authenticators faster than usual.

3.4.2 Active scanning optimization

A supplicant in the active scanning mode actively broadcasts probe request frames to search for authenticators. If the probe request frame does not have an expiration time, the authenticators will need to respond to every probe request, which causes extra traffic on the channel and degrades Layer 2 (data link layer) throughput. In contrast, FILS allows a probe request frame to include an expiration time for request replies, which means FILS can reduce the unnecessary channel scanning traffic.

3.4.3 IP address assignment during the IEEE 802.11 association

FILS provides an IP address assignment field as an information element of an authentication response frame. The assignment function enables a Dynamic Host Configuration Protocol (DHCP) server to assign an IP address to the supplicant within the Layer 2 authentication procedure. Figure 3.4 shows the frame sequence of the EAP re-authentication protocol (EAP-RP), which allows the authenticators and supplicants to process the initial link setup and IP address assignment within a few exchanged frames. This allows FILS to reduce frame exchanges in the initial link setup and shortens the setup time.

In IEEE 802.11ah networks, some papers [122, 123, 124] have proposed fast authentication procedures. These 802.11ah authentication methods mainly focus on improving the IEEE 802.11 association procedure. On the other hand, FILS not only shortens authentication procedures but also completes IP address assignment in the authentication procedure by FILS HLP Container Element.

3.4.4 Authentication information caching

FILS enables an access point and a station to cache authentication information, such as certificates, when they establish a secure link for the first time. The access point and the station use the cached authentication information to establish secure links from

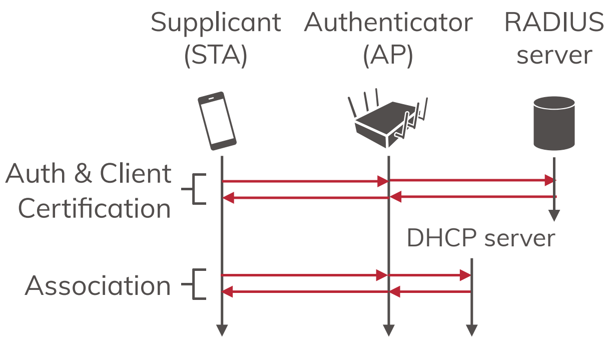

Figure 3.4: FILS frame sequences

the second time onward. This allows FILS to complete a link set up with fewer frame exchanges than a conventional EAP exchange.

3.4.5 Effectiveness

Various papers have reported the FILS performance. For example, Mano et al. performed a field experiment in a situation where 40 pedestrians with mobile devices came into a stream with the speed of 4.5 km/h and passed in the front of an access point in[105]. The mobile devices and the access point authenticated each other with FILS or WPA2-PEAP. The authors confirmed that the mobile devices using FILS established IEEE 802.11 links before the pedestrians passed the access point, while the mobile devices using WPA2-PEAP did not complete IEEE 802.11 link establishment even after passing in the front of the access point.

Ong theoretically analyzed the FILS authentication methodology outlined in IEEE P802.11 Group AI (TGai) in [106]. The author formulated the FILS active scanning enhancement and compared the performance of the FILS active scanning enhancement with IEEE 802.11 DCF (Distributed Coordination Function) or EDCA (Enhanced Distributed Channel Access). The author revealed that the FILS active scanning enhancement can make the responsiveness to beacon frames 20% and 250% faster than IEEE 802.11 EDCA and DCF, respectively. However, since this chapter refers to TGai technical papers, the FILS performance is not sufficiently analyzed with consideration of EAP-RP and IP address assignment in IEEE 802.11 association specified in the published IEEE 802.11ai standard.

Kushida et al.[107] simulated the effectiveness of FILS link setup time reduction in IEEE 802.11ad wireless networks. Their simulation results revealed that FILS reduced link setup times to ten times less than WPA2-PEAP and also reduced the number of authentication failures in an IEEE 802.11ad networks. However, all simulation nodes are stationary in their simulation scenario, and the FILS effectiveness considering the mobility of network nodes is not discussed in the study.

Although these papers showed that FILS is effective in decreasing authentication overhead, as far as I know, no paper reported has the effectiveness of link setup time reduction by FILS in vehicular networks. For this reason, this chapter presents the results of laboratory and field experiments to determine the FILS effectiveness in vehicular networks.

3.5 Experimental setup

This section describes the setup of the laboratory and field experiments conducted to evaluate the effectiveness of FILS link setup time reduction in inter-vehicular communications. The laboratory experiment was conducted to verify whether the FILS implementation can achieve the performance outlined in IEEE 802.11ai. The field experiment was conducted to verify whether FILS can work in an actual inter-vehicular communications system.

3.5.1 Details of the FILS implementation used in this study

The FILS implementation functions in ARM-based on-board units equipped with the Linux operating system (OS) because there are already a number of FILS implementations for that system. However, the FILS implementation had some differences related to IP address assignment compared to the original FILS functions due to limitations of the device driver of the IEEE 802.11 chips. Figure 3.5 shows the frame sequence of the FILS implementation. The differences between the original FILS frame sequence and the FILS implementation are shown below:

- **Authentication (RADIUS) servers work on all the access points**: Vehicles

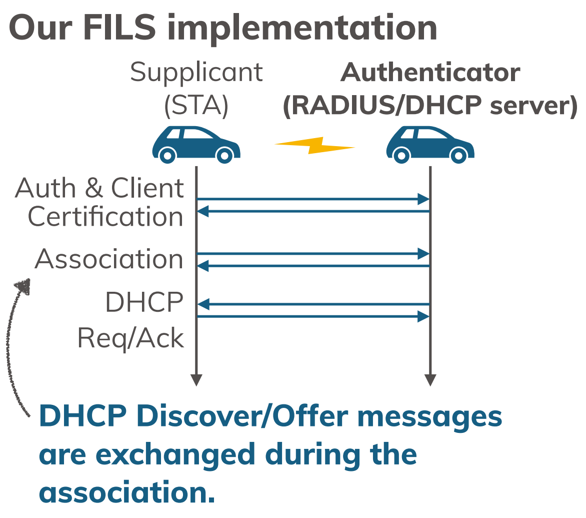
Our FILS implementation
Supplicant (STA)
Authenticator (RADIUS/DHCP server)
Auth & Client Certification
Association
DHCP Req/Ack
DHCP Discover/Offer messages are exchanged during the association.

Figure 3.5: Frame sequence of the FILS implementation used in this study

serving as DTN ferries work as authentication servers. This configuration was selected to ensure compatibility with the architecture of the disaster communication system.

- **FILS IP address assignment is not supported**: In the FILS implementation, a DHCP server assigns the client's IP address using the conventional DHCP process. In other words, the DHCP server and client must exchange DHCP discover/offer messages and request/ack messages, even though the original FILS does not require these message exchanges.

- **DHCP discover/offer messages are exchanged during the IEEE 802.11 association**: The FILS implementation supports the FILS HLP Container described in Section IV, which allows higher-layer protocols such as DHCP to send a packet during the IEEE 802.11 association. A DHCP server and client that support sending discover/offer messages via the FILS HLP Container are implemented. Therefore, the FILS implementation only needs to exchange request/ack messages after completing the Layer 2 link setup.

- **Lightweight DHCP clients work on OBUs**: The DHCP client also supports exchanging discover/offer messages via the FILS HLP Container. More specifically,

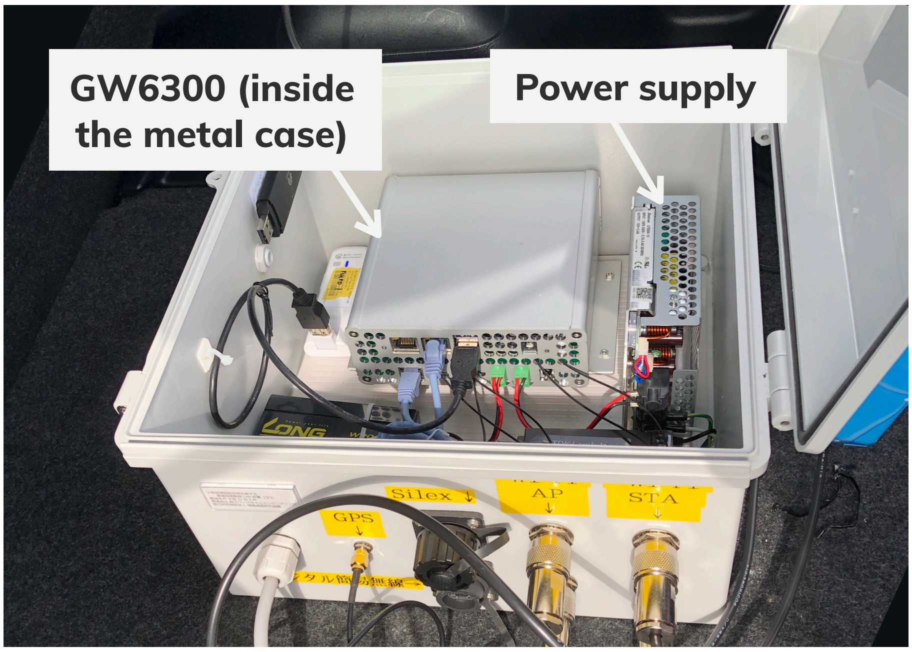

Figure 3.6: Onboard unit overview

the client sends a request message immediately after receiving an offer message via the FILS HLP Container. In contrast, a normal DHCP client will not send a request immediately because it waits for other offer messages from multiple DHCP servers.

Figure 3.6 shows the laboratory and field experiments with OBUs to measure the practical performance of the FILS implementation discussed in this chapter. Table 3.1 shows the OBU specifications. Each OBU was equipped with two Qualcomm AR9300 Wi-Fi cards and four Wi-Fi antennas that communicate over the IEEE 802.11n protocol at 2.4 GHz. Additionally, each OBU with the FILS function was equipped with a customized hostapd[90] authenticator daemon program and wireless protected access wpa_supplicant[91]. The EAP-RP function is added to the original hostapd 2.7 and wpa_supplicant 2.7 codes.

Table 3.1: Onboard unit specifications

Component	Specification
Base board	Gateworks GW6300
Linux distribution/kernel	Ubuntu 16.04.05 LTS/Linux 4.14.4
CPU	Octeon TX Dual Core ARM CPU @ 800 MHz
Memory	DDR3 1 GB
Storage	SSD 250 GB
Wi-Fi chip/driver	Qualcomm Atheros 9300 (2 chips on board) / ath9k

An embedded Structured Query Language (SQL) database is implemented in the customized hostapd to cache authentication information of stations that had previously connected to the hostapd and forced the OBUs to cache the authentication information in advance. The OBUs executes hostapd 2.6 and wpa_supplicant 2.6 without modification to measure EAP-PEAP performance levels. As described in the previous section, in both experiments, a DHCP server and client that support FILS are used.

Additionally, a RADIUS server is implemented in the authenticator, which reduced the delay between the two components to almost zero. A user application is implemented to send a file between the supplicant and the authenticator via Transmission Control Protocol (TCP). The user application sends a 100 MB file for each measurement. The kernel was allowed to reuse TCP sessions that the kernel had started and were in the TIME-WAIT state. TCP fast open[125] and Tail Loss Probe (TLP)[126] are enabled to reduce the TCP session establishment overhead. The use of TLP makes it possible to detect and recover from tail losses faster than TCP retransmission timeout. The other TCP parameters were the same as the default values.

3.5.2 Laboratory experiment configuration

I connected the OBUs with coaxial cables through variable attenuators, as shown in Figure 3.7, and manually set their signal reception strength to either -65 dBm or -95 dBm, such that the OBUs were connected at -65 dBm and disconnected at -95 dBm.

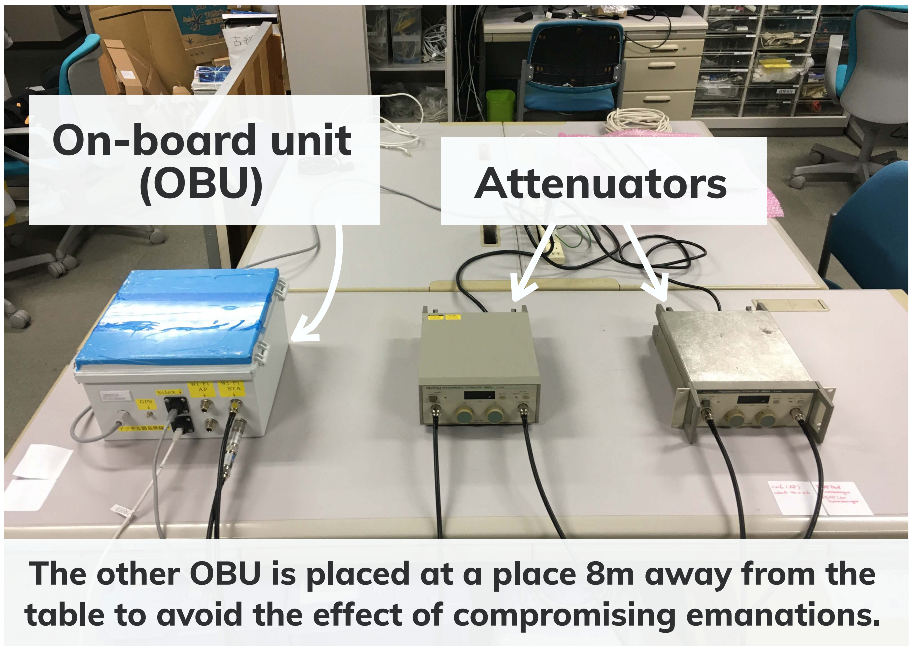
On-board unit (OBU)
Attenuators
The other OBU is placed at a place 8m away from the table to avoid the effect of compromising emanations.

Figure 3.7: Laboratory experiment setup

I also configured the data rate on Layers 2 and 1 (i.e., on both the data link layer and the physical layer) to be automatically determined. At the start of the experiment, I launched hostapd and wpa_supplicant and used the attenuators to set the receiving signal strength of the OBUs to −95 dBm. Next, I adjusted the receiving signal strength to −65 dBm and waited for 30 seconds while they attempted to transfer a 100 MB file. Finally, I restored the receiving signal strength to −95 dBm and recorded the link setup time and the number of bytes transmitted between the OBUs.

To measure link setup times, I monitored network interfaces installed at the OBUs by `ip` command[127] which is the Linux network interface utility and `iw` command[128] which is Linux WLAN configuration utility, and recorded timestamps when the IEEE 802.11 probe request is transmitted and when an IP address is assigned to a DHCP client.

The field experiment was conducted in Konan City, Kochi Prefecture, Japan. Figure 3.9 shows a map of the area and the driving route. The solid red line indicates the car trajectories while the orange dot near the center of the picture shows where I parked the car that served as the authenticator.

Two cars were used for the field experiment. The cars are referred to hereafter as

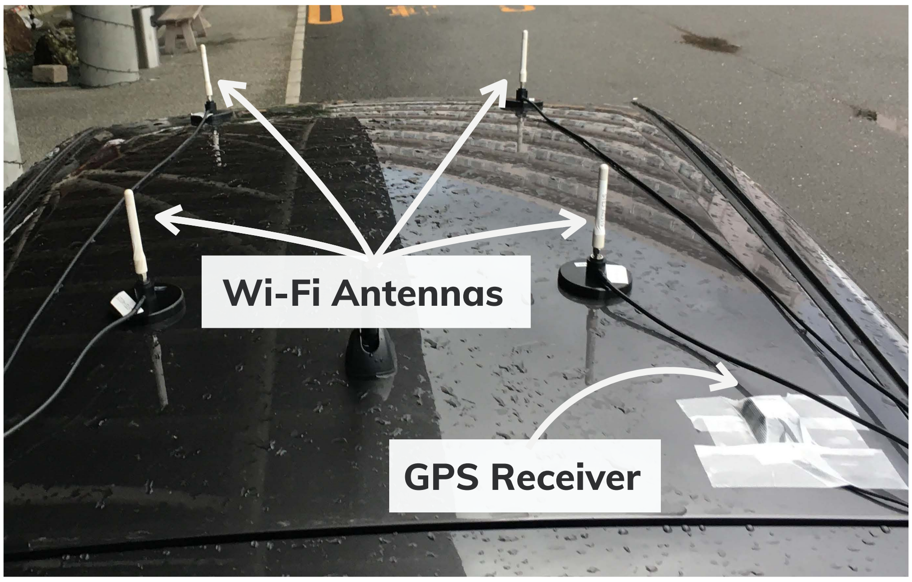

Figure 3.8: Antenna position

Car #1 and Car #2. In this experiment, Car #1 served as the authenticator, while Car #2 served as the supplicant. The OBUs and Wi-Fi antennas are installed on both cars. Figure 3.8 shows the antenna placements. The antennas of each Wi-Fi card were placed on the vehicle roofs at diagonal angles. The data rates on Layers 2 and 1 were the same as used in the laboratory experiment. The height of the cars was 1.5 m.

I moved the cars according to two scenarios, hereafter referred to as Scenario #1 and #2. In Scenario #1, I parked Car #1 at the point marked in orange in Figure 3.9 and drove Car #2 in both directions at 40 km/h on the route indicated by the red line in Figure 3.9. In Scenario #2, I drove the two cars in opposite directions at 40 km/h on the driving route so that they passed each other at the yellow point. Car #2 completed five round trips during Scenario #1, while both cars completed ten round trips during Scenario #2.

In both the laboratory and field experiments, I conducted measurements to determine if FILS shortens the link setup time and improves the amount of transmitted data. The link setup time is defined as beginning when the supplicant starts to send an association request frame to the authenticator and ending when the supplicant obtains an IP address.

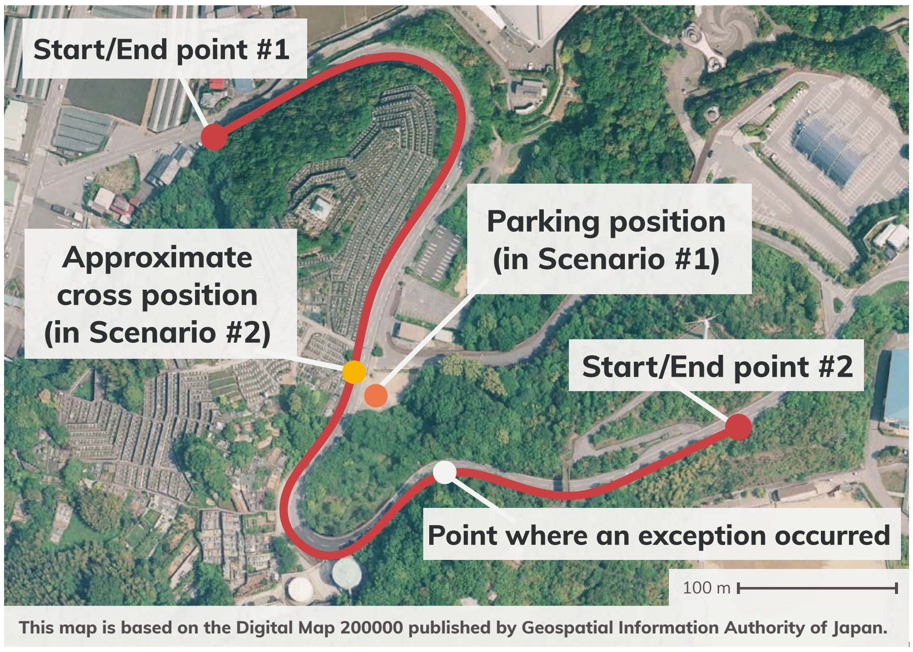
Start/End point #1
Parking position (in Scenario #1)
Approximate cross position (in Scenario #2)
Start/End point #2
Point where an exception occurred
100 m
This map is based on the Digital Map 200000 published by Geospatial Information Authority of Japan.

Figure 3.9: Field experiment area

3.6 Experiment results and discussion

This section describes the laboratory and field experimental results, which suggest the existence of a bottleneck during the initial link setup for IEEE 802.11-based inter-vehicular communications.

3.6.1 Results measured in the laboratory environment

Figure 3.10 shows a histogram comparison of the FILS and EAP-PEAP link setup times measured in the laboratory experiment. Each bin width is 0.05 s. The thin bars in deep blue and the thick bars in light blue are the FILS and EAP-PEAP results, respectively. As shown in the figures, the FILS link setup time averaged 127 ms, and the maximum and minimum setup time was 147 ms and 109 ms, respectively.

In contrast, the EAP-PEAP link setup times averaged 1.21 s with maximum and minimum setup time values of 2.81 s and 1.08 s, respectively. Since modified DHCP server and client are used, the IP address assignment with the FILS implementation was performed by DHCP after the Layer 2 link was established, as shown in Figure 3.4. Thus, the FILS

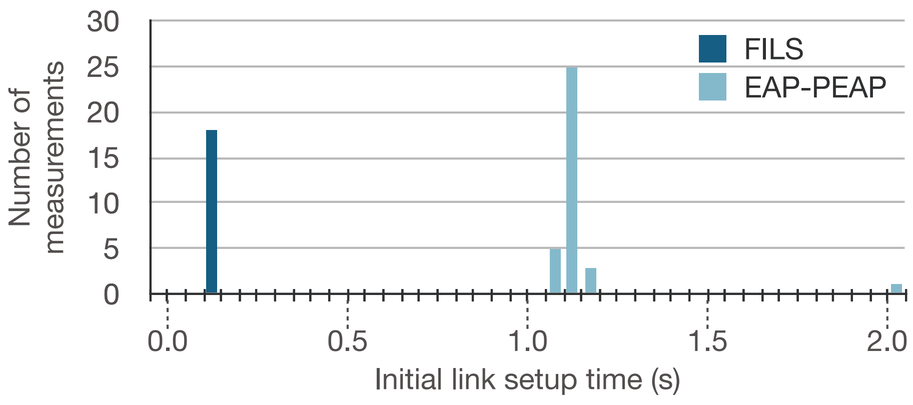

Figure 3.10: Initial link setup times measured in the laboratory experiment

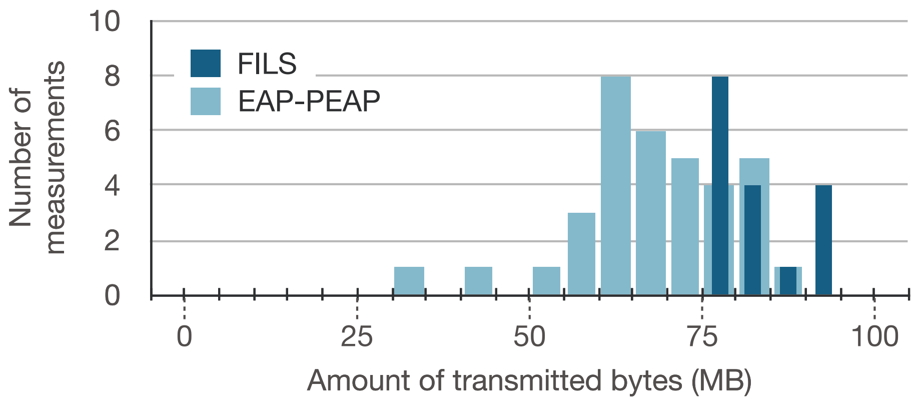

Figure 3.11: Transmitted bytes measured in the laboratory experiment

link setup times were about 30 ms longer than the link setup time of the IEEE 802.11ai standard. Figure 3.11 shows a histogram of the transmitted bytes between the OBUs in the laboratory experiment. Here, I can see that the number of bytes transmitted by FILS tends to exceed those transmitted by EAP-PEAP and that FILS increased the data traffic by about 14 MB. This result indicates that FILS reduced the link setup time and extended the communication time available to transfer data packets per connection.

From these results, I confirmed that the FILS implementation could shorten the link setup time based on the standard and increase the amount of transmitted data per connection.

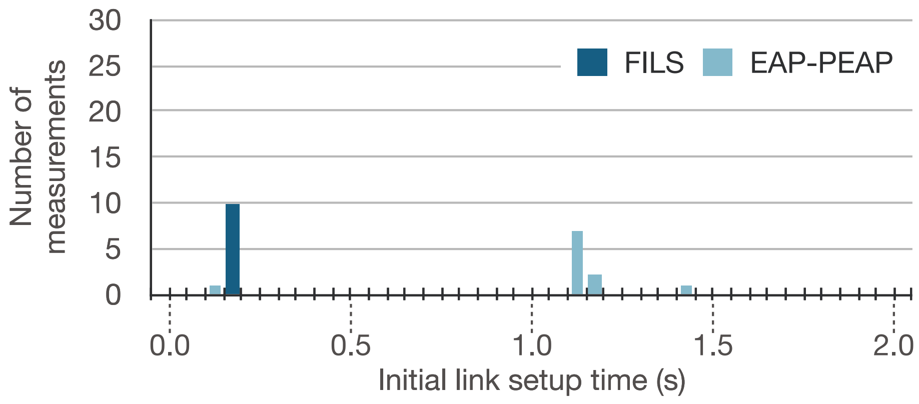

Figure 3.12: Initial link setup times measured in the field experiment (Scenario 1)

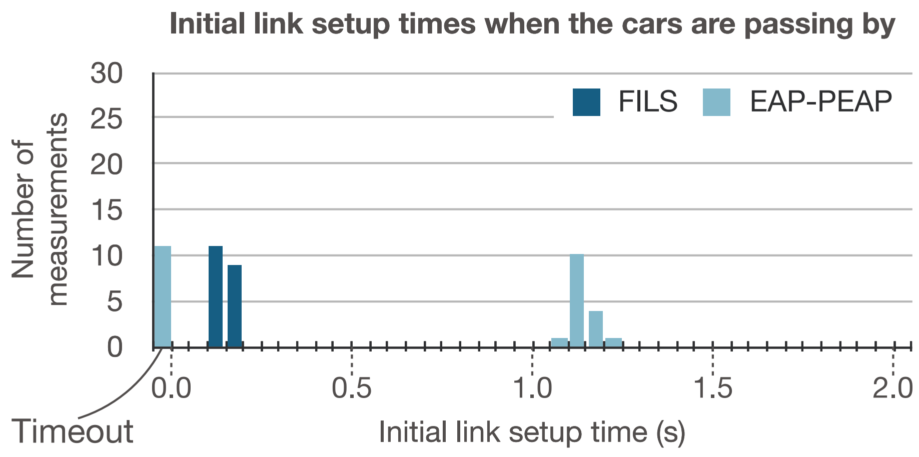

Figure 3.13: Initial link setup times measured in the field experiment (Scenario 2)

3.6.2 Results measured in the real field environment

Figure 3.12 and Figure 3.13 show histograms of the link setup times measured in the field experiments. In Scenario #1, the maximum, average, and minimum setup time of FILS were 197 ms, 173 ms, and 154 ms, respectively, while the maximum and average setup times of EAP-PEAP were 1.43 s and 1.10 s, respectively.

The minimum setup time of EAP-PEAP, except for an exceptional case, was 1.07 s. In the exceptional case, the link setup time was much shorter (137 ms) than the 1.07 s minimum because the Layer 2 link disconnected momentarily and then re-established by hostapd and wpa_supplicant without completing an EAP exchange. This occurred when

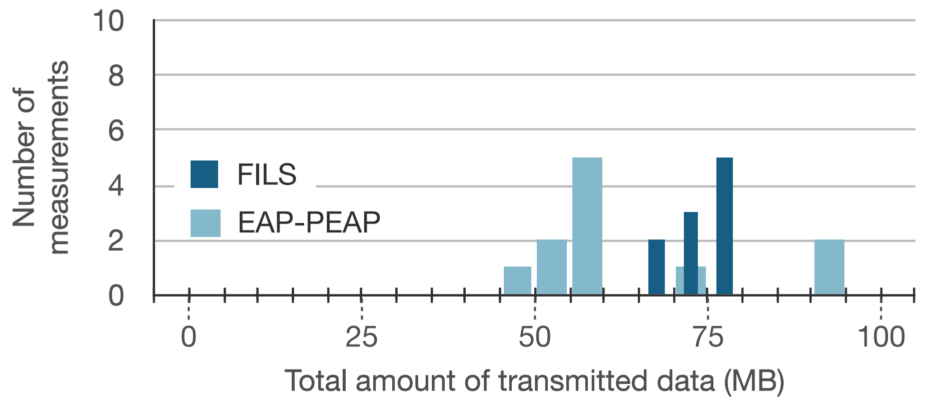

Figure 3.14: Transmitted bytes measured in the field experiment (Scenario 1)

the moving car passed at the point indicated by the white dot in Figure 3.9. At this point, the cars were visible to each other because the trees between the point and the parked car were lower than in the other areas. In Scenario #2, the maximum, average, and minimum setup times of FILS were 186 ms, 151 ms, and 110 ms, respectively, while the maximum, average, and minimum setup times of EAP-PEAP were 1.22 s, 1.13 s, and 1.09 s, respectively.

Figure 3.14 and Figure 3.15 show histograms of the transmitted bytes in the two scenarios. These results indicate that FILS increased the data traffic by around 33 MB in Scenario #1 and around 10,MB in Scenario #2. In both cases, the numbers of bytes transmitted via FILS tended to exceed those transmitted by EAP-PEAP because FILS reduced the link setup times and lengthened the time available to transfer data packets per connection.

In Scenario #1, there was one exception in which the transmitted bytes of EAP-PEAP exceeded those transmitted by FILS. This occurred when a TCP session between the cars was maintained after the link on Layer 2 had prematurely disconnected, which means the TCP session time did not expire. Normally, the user application restarts counting transmitted bytes when a TCP session is disconnected. However, in this case, the user application did not restart counting transmitted bytes because the cars established communication when Car #2 passed by the point depicted by the white dot in Figure 3.9 before passing the point depicted by the yellow dot in Figure 3.9.

Although the cars were close to each other when reaching the white point in each round

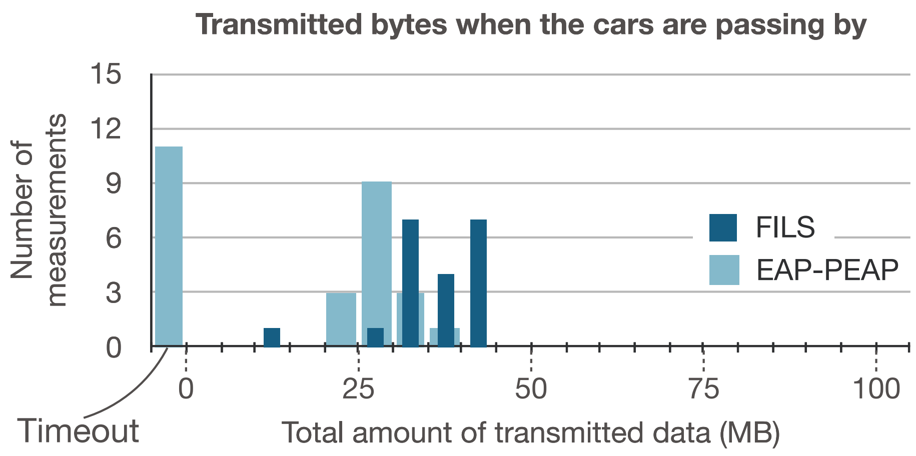

Figure 3.15: Transmitted bytes measured in the field experiment (Scenario 2)

trip, the only time the cars established a link and transferred data while passing was in the case of the exception. The Layer 2 link disconnected while Car #2 traveled between the white and yellow points, but the TCP session remained active. Because of this, the transmitted bytes were summed up before and after the Layer 2 disconnection.

In another interesting finding, FILS always established a link during Scenarios #1 and #2, while WPA2-PEAP failed to establish a link 11 times in Scenario #2. The failures of the WPA2-PEAP case occurred because the station on one car closed the IEEE 802.11 communication link with the access point on another car before sending data via TCP. In this case, the station barely received IEEE 802.11 data frames from the access point in 30 seconds after they established the communication link. I consider frame losses and the resulting failure of TCP session establishment to be the main causes of this error. In the field experiment, frame losses could occur frequently because the cars moved at the speed of 40km/h. The frame losses that stem from high mobility of cars could prevent the access point and the station from establishing a TCP session. On the other hand, FILS can set up a link with just a few of frame exchanges and prevent vehicles from missing communication opportunities due to link establishment overhead.

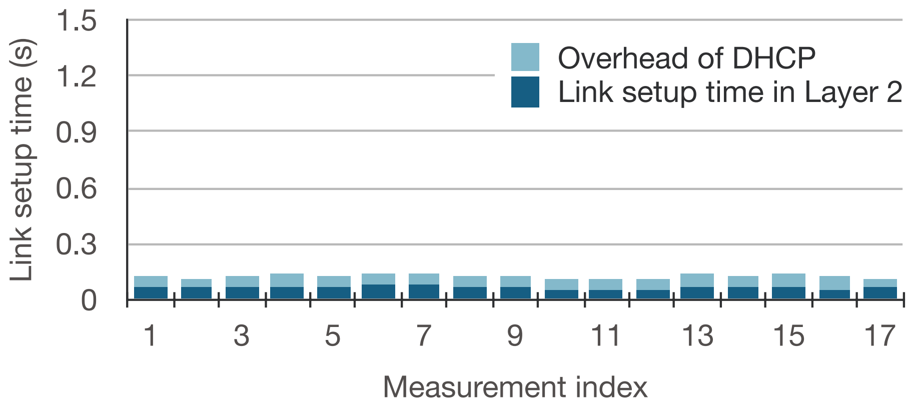

Figure 3.16: Details of FILS link setup times in the laboratory experiment

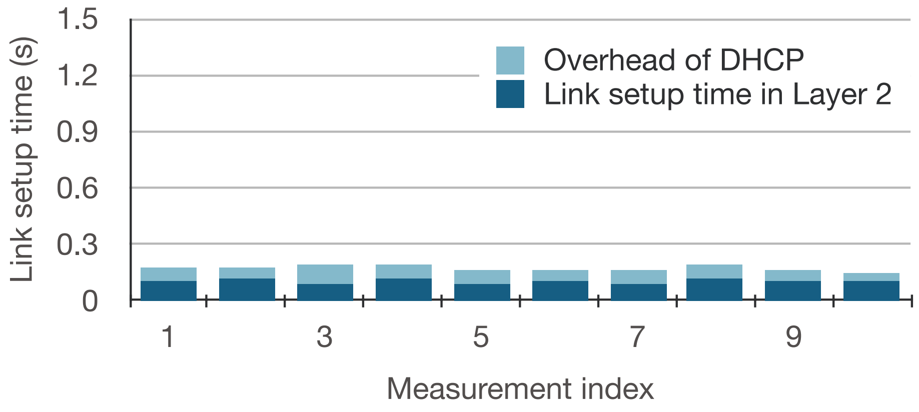

Figure 3.17: Details of FILS link setup times in the field experiment (Scenario 1)

3.6.3 Bottleneck in initial link setups

Figure 3.16 Figure 3.17, and Figure 3.18 are stacked bar graphs showing initial link setup times measured in the in-laboratory environment, Scenario #1, and Scenario #2, respectively, while Figure 3.19 Figure 3.20, and Figure 3.21 are stacked bar graphs showing the PEAP initial link setup times, respectively. Note that these figures do not include the results when the vehicles did not establish an IEEE 802.11 authentication.

From these results, I can see that FILS reduced the overhead of the client's IP address assignment by DHCP. In contrast, EAP-PEAP took approximately one second to assign the client's IP address, which indicates that the exchange of DHCP messages causes an excessive overhead.

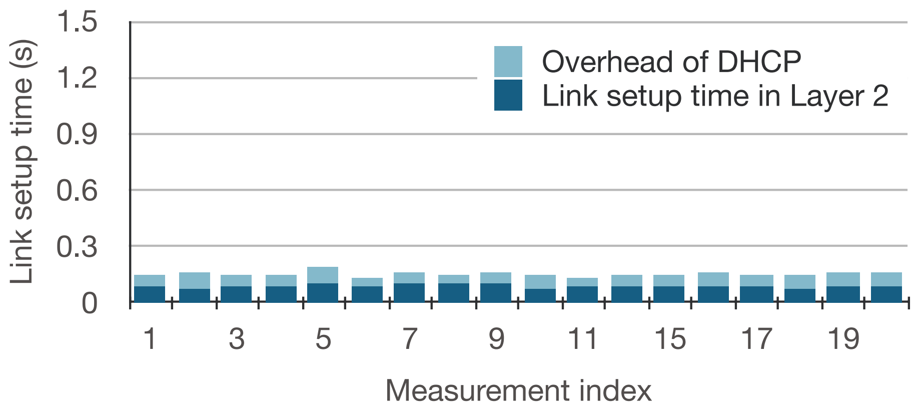

Figure 3.18: Details of FILS link setup times in the field experiment (Scenario 2)

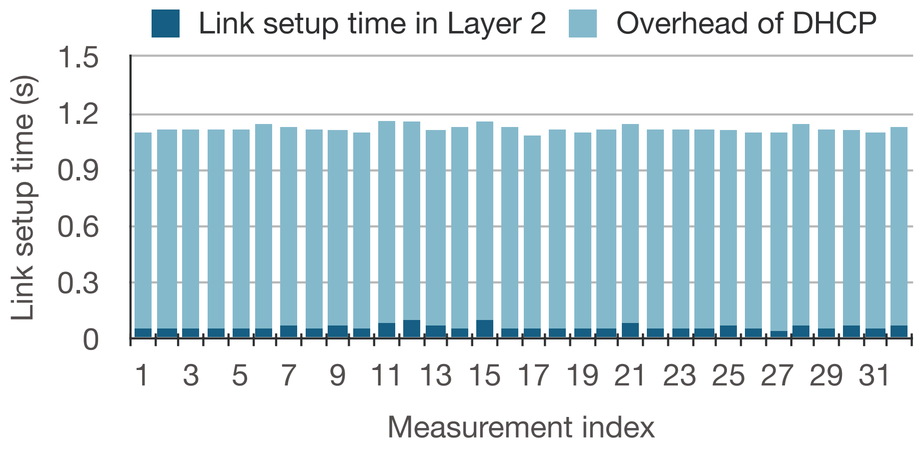

Figure 3.19: Details of WPA2-PEAP link setup times in the laboratory experiment

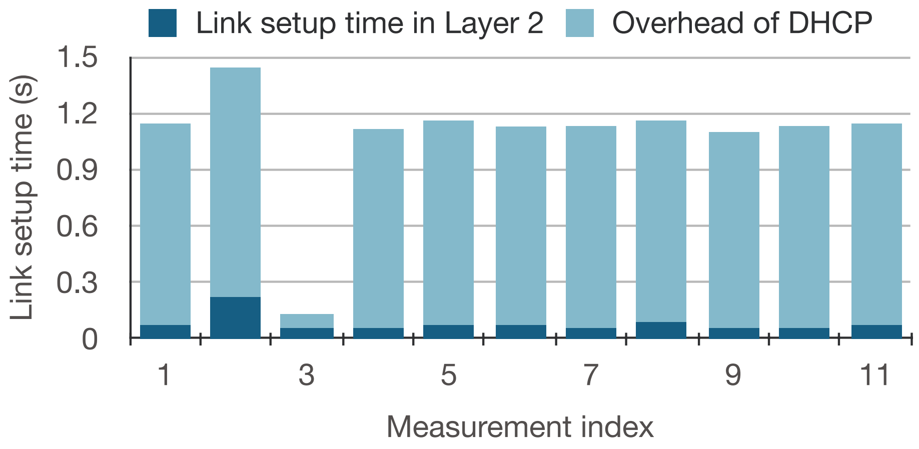

Figure 3.20: Details of WPA2-PEAP link setup times in the field experiment (Scenario 1)

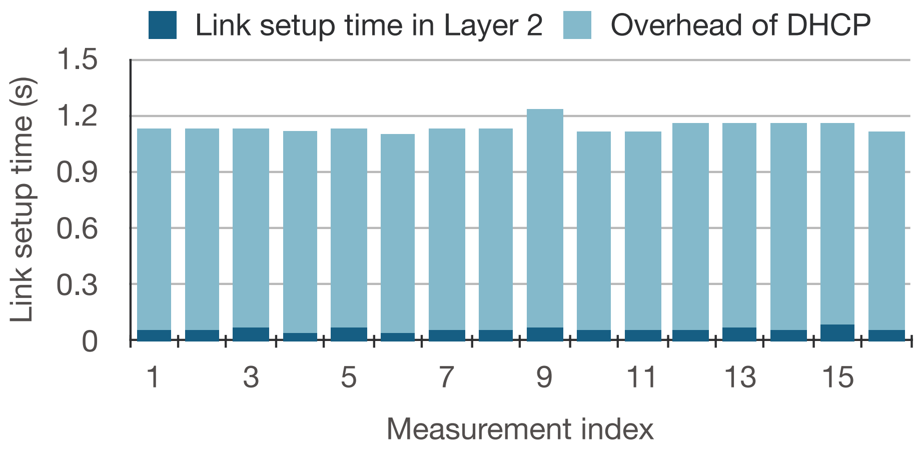

Figure 3.21: Details of WPA2-PEAP link setup times in the field experiment (Scenario 2)

There are two potential reasons for the DHCP overhead: DHCP offer message waiting time and parsing DHCP lease files. A DHCP client waits for several seconds before responding to a DHCP offer message because it can receive these messages from multiple DHCP servers. In addition, for a major DHCP server with full DHCP implementation, such as an ISC DHCP server, records assign IP addresses in a lease file and check this file prior to each assignment to validate the IP address lease time and to avoid conflicts between assigned IP addresses. Consequently, there is a delay due to having to parse this lease file.

However, in the experiments, the delay from parsing the lease file can be ignored because I remove the lease file and reset the DHCP server prior to each measurement. Thus, unlike in the conventional case, the FILS implementation discussed in this study allows the DHCP server to send an offer message during the IEEE 802.11 association using the FILS HLP Container, upon receipt of which, the client immediately sends immediately a request message back via the FILS HLP Container. As a result, the DHCP overhead of the proposed FILS implementation is shorter than that of EAP-PEAP. In Figure 3.20, I see an exceptional case, whose index is three, in which the EAP-PEAP link setup time is shorter than in other cases. This exceptional case is due to the same reason that the cars completed the link setup time over the low trees area. In still another case, a situation occurred in which EAP-PEAP took a longer time to set up a link than the other situations, but the cause of that exception was traced to frame losses, which exceeded those of other cases.

These results show that EAP-PEAP could complete the Layer 2 link as quickly as FILS, but the results do not include the overhead of certificate verification to the authentication server. Therefore, the EAP-PEAP link setup could take longer in situations where the authentication server is accessible over other networks such as cellular networks. Furthermore, these results do not include the overhead that occurs when a DHCP server checks lease files, which could harm the initial link setup when numerous vehicles attempt to connect with each other.

However, the results described above confirm that FILS reduced link setup times and increased the size of transmitted data between two passing cars communicating over IEEE 802.11n with WPA2-EAP. Additionally, FILS was found to be capable of quickly establishing secure links in IEEE 802.11-based VDTNs with the same level of security as WPA2-EAP as well as preventing vehicles from missing communication opportunities when they pass each other.

I consider two reasons for the communication errors when using WPA2-PEAP: frame loss and the failure of TCP session establishment. Since the cars passed by each other at high speed, and frame loss easily occurred, the data transfer applications could not establish a TCP session. In addition, the frame loss could prevent the cars from transferring the file after the TCP session was established. For this reason, the station did not receive data frames and closed a communication link on Layer 2 intentionally. The communication errors I observed in the field experiment indicate that data transmission over TCP can fail when a connection on Layer 2 and lower is unstable even if the connection on Layer 2 is established successfully. UDP can be an alternative protocol of TCP, but UDP does not have a retransmission mechanism. Therefore, a cross-layer mechanism that can monitor the connection state of each network layer (e.g., TCP and user applications) and notify the upper layers like TCP and user applications of the connection states is required to avoid frame losses and transfer large amounts of data between vehicles.

3.7 Conclusions

This chapter presented the first empirical investigation showing that FILS (IEEE,802.11ai) reduces link setup times and increases the size of transferred application data in 2.4 GHz IEEE 802.11n-based inter-vehicular communications. More specifically, when cars passed each other at a relative speed of 80 km/h, the results showed that FILS reduced the initial link setup to around 150 ms between the passing cars, and that it transferred around 40 MB, which is 10 MB more than WPA2-PEAP. This study also confirmed that FILS prevented vehicles from missing communication opportunities due to the link establishment overhead.

Chapter 4 Conclusions

This dissertation has focussed on developing vehicular network systems with consideration of the deployment to the real world and developed a wireless LAN emulator with wireless network tap devices (namely WiNE-Tap) and investigated the practical performance of initial link setup and bulk data transmission in intermittent inter-vehicular communication based on field experiments. This chapter concludes the studies, discuss the utilization of the contributions of this dissertation for developing vehicular network systems, and at last show the prospects of vehicular network technologies.

4.1 Summary of this dissertation

Vehicular networking is a significant technology for realizing the intelligent transportation system (ITS) and is expected to enhance road services by sharing location information, sensor information, photos and videos among vehicles. Vehicular networking also can improve communication systems under natural disaster conditions; for example, emergency vehicles equipped with onboard units keeps connectivity between disaster-stricken areas without any communication links by carrying data physically.

Since the vehicular network systems are expected to improve road safety and disaster communication must operate without malfunctions to avoid harming road safety and disturbing disaster response, the developers and researchers of the vehicular network systems must be responsible for meeting requirements: reliable implementation, which the implementation of the vehicular network systems work reliably, and security to protect the vehicular network systems from malicious users. However, insufficient development environments for vehicular networks and not enough empirical investigations about the inter-vehicular bulk data transmission make it difficult to develop the vehicular network systems and validate their operation, and prevent the vehicular networks deployment to the real world. Thus, this dissertation proposed the wireless network emulator with wireless network tap devices (WiNE-Tap) that can be used for the operation validation system for IEEE 802.11 network applications and protocol stacks with actual implementation and presented the first empirical investigation of link setup time and bulk data transmission performance in inter-vehicular communication with comparison of FILS and WPA-PEAP.

■ Wireless LAN emulator with wireless network tap devices

Wireless network emulation is useful to validate the operation of the vehicular network systems. Wireless network emulation partly abstracts the behavior of the network systems such as radio propagation and network nodes' mobility and enables the applications of the wireless network systems to operate in a virtual network in which a network simulator imitates the behavior of the vehicular network environment such as vehicles' mobility and radio propagation. Therefore, it allows users to reproduce the behavior of the network applications as they perform in the real network environment. Although developing the vehicular network systems requires validating the operation of both network applications and protocol stacks, the existing wireless network emulators focus on link emulation only. These network emulators capture data packet flow through virtual network devices such as TUN/TAP devices and apply bandwidth restriction and transmission delay insertion to the captured data packet flow based on radio propagation and mobility models. Meanwhile, they do not allow the network protocol stacks such as IEEE 802.11 protocols to operate in the virtual network. For this reason, it makes it difficult to develop vehicular network systems, which require developers and researchers to check the protocol implementation for meeting the requirements for the vehicular network systems.

The wireless LAN emulator with wireless network tap devices (WiNE-Tap) allows the IEEE 802.11 protocol implementation for Linux systems such as `cfg80211` and `mac80211` and network applications to operate in a virtual network, which the network simulator simulates the behavior of radio propagation and network nodes' mobility. WiNE-Tap provides the wireless network emulation framework to connect the IEEE 802.11 protocol implementation for Linux systems and a user application such as a network simulator and enables them to exchange IEEE 802.11 frames and IEEE 802.11 device configuration parameters such as transmission power and received signal strengths. In addition, WiNE-Tap is designed independently from a specific network simulator and realizes a wireless LAN emulation environment with a network simulator that users want to use. The existing wireless network emulators are designed for a specific network simulator. Therefore, WiNE-Tap allows the IEEE 802.11 protocol implementation and the network simulator to conduct link emulation using real IEEE 802.11 frames and reproduce the behavior of the IEEE 802.11 protocol implementation when the channel conditions such as received signal strengths dynamically change. The performance evaluation shows the IEEE 802.11 protocol implementation for Linux systems and network applications operate in a virtual network simulated by a network simulator, Scenargie, and can be used as the operation validation system for vehicular network systems.

This dissertaiton showed the performance evaluation of WiNE-Tap when two wireless nodes connect over an wireless link of IEEE 802.11a, which is the base of IEEE 802.11p standard designed for V2X communications. The performance evaluation results revealed the WiNE-Tap implementation described in the dissertation emulate throughputs with up to 6 Mbps between the two nodes. The performance is at least sufficient to emulate the behavior of vehicular network systems in which a vehicle broadcasts a basic safety message with up to a few of hundreds bytes payload every 100ms.

■ Initial link setup time reduction and bulk data transmission improvement performance by FILS for vehicular networks

The conventional security mechanism for vehicular networks such as IEEE 1609.2 and IEEE 802.11i/IEEE 802.11 provides enterprise authentication based on certificates. However, it can degrade the performance in transferring applications data in intermittent inter-vehicular communication because they require message exchanges to share encryption keys safely and take a few seconds to establish a secure communication link between the vehicles. For this reason, it is necessary to reduce the overhead of the initial link setup between the vehicles. Fast Initial Link Setup (FILS) is a security protocol for IEEE 802.11 and reduces the authentication overhead because of cached authentication information and allows an access point and a station of IEEE 802.11 networks to establish a secure link as secure as WPA2 Enterprise authentication within 100 ms. Therefore, this paper proposed applying FILS for vehicular networks' authentication mechanism and reducing the authentication overhead in intermittent inter-vehicular communication.

The performance evaluation shown in this paper reveals that FILS allows the vehicles passing by with high speed to establish a secure IEEE 802.11 communication link within 130 ms and benefits to mitigate the authentication overhead in intermittent inter-vehicular communication. In addition, the link setup time reduction by FILS increases the amount of data transferred between the vehicles than WPA2-PEAP. The performance evaluation of WPA2-PEAP shows that WPA2-PEAP degrades the authentication overhead between the passing vehicles and prevents the vehicles from completing the link establishment because of frame losses.

4.2 Future direction

There are several ways to improve the performance of WiNE-Tap. Since the network simulator used for the performance evaluation frequently referred to the real-time system clock to strictly emulate the IEEE 802.11 frame transmission and reception, the system clock should respond quickly to reduce the delay to wait for the system clock reference, or the network simulator should reduce the number of references to the clock. The network simulator discards IEEE 802.11 frames from `wtap80211` when the frame transmission and reception events are not ignited at the scheduled times due to the processing overhead of the other simulation events or the delays to wait for the system clock response. To avoid discarding the frames, using the real-time kernel and another computer with higher performance than the computers used in this paper will be able to increase the performance.

The real-time kernel works with shorter CPU ticks than the general-purpose kernel. For example, the Linux real-time kernel (namely, the low-latency kernel) CPU tick is configured to 1000 Hz, which is 10 times more than the CPU ticks of the Linux general-purpose kernel. Thus, the real-time kernel can process the system calls 10 times faster than the general-purpose kernel. Therefore, the WiNE-Tap performance can be improved by using the real-time kernel. The high-performance computer also possibly improve the responsiveness of the system clock and enhance the processing ability of the simulation/emulation host machines.

The performance evaluation of WiNE-Tap with other network simulators such as ns-3 and OMNeT++ should be performed because they have different implementations from the network simulator used in this paper. For example, ns-3 provides the direct code execution (DCE)[46], an extension module for network emulation. Since its original release, DCE has been periodically updated and enables link emulation with high bitrates about tens of megabytes per seconds[46]. However, it should be confirmed how strictly the emulation results of WiNE-Tap with a different network simulator are reproduced. The strictness of the emulation results and the maximum throughput that can be emulated on WiNE-Tap has the trade-off relationship because the number of simulation events such as radio propagation increases as the network simulator reproduces the behavior of the network system more strictly.

The paper showed the architecture and an example implementation of WiNE-Tap. The performance evaluation results of WiNE-Tap shown in this paper validated the operation and performance of an implementation of WiNE-Tap with generally used network applications not only used for vehicular networks. Performance evaluation with real vehicular network applications will be required to validate and improve WiNE-Tap functionality for vehicluar network emulation.

On the link setup time and bulk data transmission investigation, although the laboratory and field experiments revealed that the FILS initial link setup time reduction effectively reduces the authentication overhead in inter-vehicular communication, the authentication overhead can be reduced more by using another FILS implementation that fully follows the original standards.

FILS implementation used in the investigation does not support some functions of the original FILS standards due to the device driver. For example, the FILS implementation does not support active scanning. In the field experiment, the vehicles passed by on a straight road and had a line of sight. Therefore, one vehicle had enough time to detect beacon frames from the other vehicle before they passed by. However, if the vehicles had passed by on a curve and had not had a line of sight before passing by, they could not detect the beacon frames and possibly delay beginning the authentication. Active scanning allows the vehicles to detect each other more quickly and begin the authentication procedure, reducing the authentication overhead. In addition, the FILS implementation requires an access point and a station to exchange frames one more than the original FILS. The original FILS standard enables the IP address assignment by DHCP in IEEE 802.11 association/authentication frame exchanges. For this reason, the practical performance of FILS in vehicular networks can be improved by using the FILS implementation following the original one. Whereas this paper presented the FILS performance when one vehicle authenticates the other vehicle while the vehicles were passing by each other, the effectiveness of FILS link setup time reduction when multiple vehicles connect to the access point simultaneously should be performed to clarify how the number of concurrent connections to an access point in inter-vehicular communications can be increased by FILS.

4.3 Prospects of vehicular networks

The contributions of this paper will create a stir to the development activity of vehicular network systems, considering deploying them in the real world. Various researchers and organizations have indicated the usability of vehicular networks for years[129, 130], but they have not been deployed or popularized in the real world yet. This study can break the standstill of developing vehicular network systems.

The vehicular network systems require to break various barriers to be deployed, such as the innovation of vehicular communication technologies, legislation, and the understandings of vehicular networks, etc. In the U.S., the Federal Communication Committee (FCC) assigned the 75 MHz channel width in 5.9 GHz frequency bands for Dedicated Short Range Communication (DSRC), i.e., IEEE 802.11p, in 1999[131], and the vehicular network can be deployed if the communication technologies and implementation are ready. However, there are a few of research about the development of vehicular network systems considering deploying in the real world, and there is a gap between the users and developers of vehicular network systems. The researchers and developers of vehicular networks see ahead of their work, but unfortunately, their work is still not reduced to the society with concrete benefits. The gap at least comes from the insufficient concrete discussion on the development methodology of vehicular network systems.

Vehicular network systems cannot be deployed in the real world without the understandings of people because they can closely affect road safety. Conventionally, it is challenging to validate the operation of vehicular network systems due to the difficulty of performing field testing and strict technical requirements. For being acceptable from people, the developers and researchers of vehicular networks should show people sufficient grounds to prove the safety with specific operation validation methods and results and benefits of the vehicular network systems. However, the many kinds of research in vehicular networking are evaluated based on network simulation and aim to prove the performance of their proposed methods. They insufficiently have the consideration to deploy the vehicular network systems in the real world. For encouraging to popularize vehicular network systems in the real world, it is necessary to conduct the performance evaluation based on simulation and implementation. The contributions of this paper showed an example of the operation validation method and empirical measurement results for vehicular network systems with implementation and will contribute to the research and development community to encourage their activities.

Acknowledgement

I would like to thank Prof. Susumu Ishihara, a professor of the college of engineering, Shizuoka university, for graceful instruction over the whole of the study.

I would like to appreciate the cooperation of the co-authors of the papers related to the dissertation: Mr. Taka Maeno and Dr. Mineo Takai of Space-Time Engineering, LLC., Dr. Yasunori Owada, Dr. Goshi Sato, Dr. Katsuhiro Temma, and Dr. Toshiaki Kuri of National Institute of Communications and Technology (NICT).

I would like to thank to Prof. Akihiro Sugiura of the college of informatics, Shizuoka university, for undertaking the chair of the examination committee and giving the thoughtful comments. In addition, I would like to thank to Prof. Hiroshi Mineno of the college of informatics, Shizuoka University, and Prof. Satoru Morita of the college of engineering, Shizuoka university, for undertaking examination committees and giving thoughtful comments.

I would like to appreciate economical, emotional supports from my family to accomplish the study and this dissertation.

This work is derived from the studies partly supported by JSPS KAKENHI 19H04092, 15H02689, 17K20027.

This work is also derived from the study supported by Council for Science, Technology and Innovation (CSTI), Cross-ministerial Strategic Innovation Promotion Program (SIP) 2nd, “Enhancement of National Resilience against Natural Disaster” with funding from the National Research Institute for Earth Science and Disaster Resilience (NIED).

References

[1] ETSI EN 302 637-2 v1.4.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.04.01_60/en_30263702v010401p.pdf. (Accessed at 2021-10-29).

[2] David Eckhoff, Nikoletta Sofra, and Reinhard German. A performance study of cooperative awareness in ETSI ITS G5 and IEEE WAVE. In *2013 10th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 196–200, Banff, AB, Canada, March 2013. IEEE. http://ieeexplore.ieee.org/document/6578347/.

[3] Hendrik-Jörn Günther, Björn Mennenga, Oliver Trauer, Raphael Riebl, and Lars Wolf. Realizing collective perception in a vehicle. In *2016 IEEE Vehicular Networking Conference (VNC)*, pp. 1–8, December 2016.

[4] David Eckhoff, Nikoletta Sofra, and Reinhard German. A performance study of cooperative awareness in ETSI ITS G5 and IEEE WAVE. In *2013 10th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 196–200, March 2013.

[5] Alessandro Bazzi, Barbara M. Masini, Alberto Zanella, and Ilaria Thibault. On the Performance of IEEE 802.11p and LTE-V2V for the Cooperative Awareness of Connected Vehicles. *IEEE Transactions on Vehicular Technology*, Vol. 66, No. 11, pp. 10419–10432, November 2017.

[6] Hendrik-Jörn Günther, Björn Mennenga, Oliver Trauer, Raphael Riebl, and Lars Wolf. Realizing collective perception in a vehicle. In *2016 IEEE Vehicular Networking Conference (VNC)*, pp. 1–8, December 2016.

[7] D. de Bruin, J. Kroon, R. van Klaveren, and M. Nelisse. Design and test of a cooperative adaptive cruise control system. In *IEEE Intelligent Vehicles Symposium, 2004*, pp. 392–396, June 2004.

[8] Arturo Davila and Mario Nombela. Platooning - Safe and Eco-Friendly Mobility. In *SAE 2012 World Congress & Exhibition*, pp. 2012–01–0488, April 2012. `https://www.sae.org/content/2012-01-0488/`.

[9] Steven E. Shladover. PATH at 20—History and Major Milestones. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 8, No. 4, pp. 584–592, December 2007.

[10] Francesco Malandrino, Claudio Casetti, Carla-Fabiana Chiasserini, and Marco Fiore. Content downloading in vehicular networks: What really matters. In *2011 Proceedings IEEE INFOCOM*, pp. 426–430, April 2011.

[11] DSSS [Driving Safety Support Systems] — Functions and Services of UTMS — UTMS Society of Japan. `https://www.utms.or.jp/english/system/dsss.html`.

[12] Toyota Safety Technology ITS Connect (in Japanese). `https://toyota.jp/technology/safety/itsconnect/`. (Accessed at 2021-10-29).

[13] Uichin Lee, Biao Zhou, Mario Gerla, Eugenio Magistretti, Paolo Bellavista, and Antonio Corradi. Mobeyes: Smart mobs for urban monitoring with a vehicular sensor network. *IEEE Wireless Communications*, Vol. 13, No. 5, pp. 52–57, October 2006.

[14] Arata Kato, Taka Maeno, Yasunori Owada, Goshi Sato, Katsuhiro Temma, Toshiaki Kuri, Mineo Takai, and Susumu Ishihara. Link Setup Time Reduction by FILS on IEEE 802.11-based Inter-vehicular Communications. *IEEE Access*, Vol. 9, pp. 159796–159808, 2021. `https://ieeexplore.ieee.org/document/9618946/`.

[15] *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as Amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009): IEEE Standard for Information Technology– Local and Metropolitan.* IEEE, Place of publication not identified, 2010. http://ieeexplore.ieee.org/servlet/opac?punumber=5514473.

[16] ETSI 3rd Generation Partnership Project (3GPP). ETSI TS 123 285 V14.3.0 - Universal Mobile Telecommunications System (UMTS); LTE; Architecture enhancements for V2X services (3GPP TS 23.285 version 14.3.0 Release 14). https://www.etsi.org/deliver/etsi_ts/123200_123299/123285/14.03.00_60/ts_123285v140300p.pdf, 2017. (Accessed at 2021-10-29).

[17] IEEE P802.11 - TASK GROUP BD (NGV) - GROUP INFORMATION UPDATE. https://www.ieee802.org/11/Reports/tgbd_update.htm. (Accessed at 2021-10-29).

[18] ETSI 3rd Generation Partnership Project (3GPP). ETSI TS 138 211 v16.2.0 - 5G; NR; Physical channels and modulation (3GPP TS 38.211 version 16.2.0 Release 16). https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/16.02.00_60/ts_138211v160200p.pdf, 2020.

[19] ETSI 3rd Generation Partnership Project (3GPP). ETSI TS 138 214 v16.2.0 - 5G; NR; Physical layer procedures for data (3GPP TS 38.214 version 16.2.0 Release 16). https://www.etsi.org/deliver/etsi_ts/138200_138299/138214/16.02.00_60/ts_138214v160200p.pdf, 2020.

[20] A Betsy Felicia and L Lakshmanan. Survey on Accident Avoidance and Privacy Preserving Navigation System in Vehicular Network. *Global Journal of Pure and Applied Mathematics*, Vol. 12, No. 1, pp. 943–949, 2016.

[21] Mohammed Saad Talib. Converging VANET with Vehicular Cloud Networks to reduce the Traffic Congestions: A review. Vol. 12, No. 21, p. 9, 2017.

[22] Paolo Bellavista, Eugenio Magistretti, Uichin Lee, and Mario Gerla. *Standard Integration of Sensing and Opportunistic Diffusion for Urban Monitoring in Vehicular Sensor Networks: The MobEyes Architecture.* July 2007.

[23] U.S. Department of Transportation. Connected Vehicle Benefits. https://www.its.dot.gov/factsheets/pdf/ConnectedVehicleBenefits.pdf.

[24] Ho Ting Cheng, Hangguan Shan, and Weihua Zhuang. Infotainment and road safety service support in vehicular networking: From a communication perspective. *Mechanical Systems and Signal Processing*, Vol. 25, No. 6, pp. 2020–2038, August 2011. https://www.sciencedirect.com/science/article/pii/S0888327010004127.

[25] Insick Son, Kabsu Han, Daejin Park, Meng Di Yin, and Jeonghun Cho. A Study on Implementation of IVI Applications for Connected Vehicle Using HTML5. In *2014 International Conference on IT Convergence and Security (ICITCS)*, pp. 1–4, October 2014.

[26] Cabinet Office, Government of Japan. The 5th Science and Technology Basic Plan (Society 5.0).

[27] U.S. Department of Transportation. Smart City Challenge. https://www.transportation.gov/sites/dot.gov/files/docs/Smart%20City%20Challenge%20Lessons%20Learned.pdf. (Accessed at 2021-10-29).

[28] European Commission. Directorate General for Research and Innovation. *Industry 5.0: Towards a Sustainable, Human Centric and Resilient European Industry.* Publications Office, LU, 2021. https://data.europa.eu/doi/10.2777/308407.

[29] Chinese Communist Party. Made in China 2025 (English version translated by IoT-ONE). http://www.cittadellascienza.it/cina/wp-content/uploads/2017/02/IoT-ONE-Made-in-China-2025.pdf. (Accessed at 2021-10-29).

[30] APT Recommendation on Specification of Information and Communication System Using Vehicle During Disaster, 2018.

[31] Connected Vehicle technology is coming to the streets of New York City!. — NYC Connected Vehicle Project. https://www.cvp.nyc/. (Accessed at 2021-10-29).

[32] THEA Connected Vehicle Pilot. https://theacvpilot.com/. (Accessed at 2021-10-29).

[33] Wyoming DOT Connected Vehicle Pilot. https://wydotcvp.wyoroad.info/. (Accessed at 2021-10-29).

[34] V2X Core Technical Committee. On-Board System Requirements for V2V Safety Communications. Technical report, SAE International. https://www.sae.org/content/j2945/1_202004.

[35] Arata Kato, Mineo Takai, and Susumu Ishihara. WiNE-Tap: Wireless LAN emulator with wireless network TAP devices. *Ad Hoc Networks*, p. 102690, September 2021. https://linkinghub.elsevier.com/retrieve/pii/S1570870521001943.

[36] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wiessner. Microscopic Traffic Simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2575–2582, November 2018.

[37] Miguel Baguena, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. Towards realistic vehicular network simulation models. In *2012 IFIP Wireless Days*, pp. 1–3, Dublin, Ireland, November 2012. IEEE. http://ieeexplore.ieee.org/document/6402805/.

[38] Le Wang, Renato Iida, and Alexander M. Wyglinski. Vehicular Network Simulation Environment via Discrete Event System Modeling. *IEEE Access*, Vol. 7, pp. 87246–87264, 2019.

[39] Mani Amoozadeh, Bryan Ching, Chen-Nee Chuah, Dipak Ghosal, and H. Michael Zhang. VENTOS: Vehicular Network Open Simulator with Hardware-in-the-Loop Support. *Procedia Computer Science*, Vol. 151, pp. 61–68, 2019. https://linkinghub.elsevier.com/retrieve/pii/S1877050919304739.

[40] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, Vol. 10, No. 1, pp. 3–15, January 2011.

[41] nsnam. Ns-3. https://www.nsnam.org/. (Accessed at 2021-10-29).

[42] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: Rapid prototyping for software-defined networks. pp. 1–6. ACM Press, 2010. http://portal.acm.org/citation.cfm?doid=1868447.1868466.

[43] Mineo Takai, Jay Martin, Shigeru Kaneda, and Taka Maeno. Scenargie as a Network Simulator and Beyond. *Journal of Information Processing*, Vol. 27, No. 0, pp. 2–9, 2019. https://www.jstage.jst.go.jp/article/ipsjjip/27/0/27_2/_article.

[44] EXata® Network Emulator Software. https://www.scalable-networks.com/products/exata-network-emulator-software/. (Accessed at 2021-10-29).

[45] Universal TUN/TAP device driver — The Linux Kernel documentation. https://www.kernel.org/doc/html/v5.12/networking/tuntap.html. (Accessed at 2021-10-29).

[46] Hajime Tazaki, Frédéric Uarbani, Emilio Mancini, Mathieu Lacage, Daniel Camara, Thierry Turletti, and Walid Dabbous. Direct code execution: Revisiting library OS architecture for reproducible network experiments. pp. 217–228. ACM Press, 2013. http://dl.acm.org/citation.cfm?doid=2535372.2535374.

[47] Intrig-unicamp/mininet-wifi. https://github.com/intrig-unicamp/mininet-wifi, November 2021.

[48] Tc(8) - Linux manual page. https://man7.org/linux/man-pages/man8/tc.8.html. (Accessed at 2021-10-29).

[49] Cozybit/wmediumd. https://github.com/cozybit/wmediumd, September 2019. (Accessed at 2021-10-29).

[50] Empowering App Development for Developers — Docker. https://www.docker.com/. (Accessed at 2021-10-29).

[51] QEMU. https://www.qemu.org/. (Accessed at 2021-10-29).

[52] Stephen Hemminger. Network Emulation with NetEm, 2005.

[53] Marta Carbone and Luigi Rizzo. Dummynet revisited. *ACM SIGCOMM Computer Communication Review*, Vol. 40, No. 2, pp. 1–12, April 2010. `http://portal.acm.org/citation.cfm?doid=1764873.1764876`.

[54] Johan Garcia, Emmanuel Conchon, Tanguy Pérennou, and Anna Brunstrom. KauNet: Improving Reproducibility for Wireless and Mobile Research. In *MobiEval 2007*, pp. p.21–26, Puerto Rico, United States, June 2007. `https://hal.archives-ouvertes.fr/hal-00388795`.

[55] Tanguy Pérennou, Emmanuel Conchon, Laurent Dairaine, and Michel Diaz. Two-Stage Wireless Network Emulation. In Thierry Gayraud, Michel Mazzella, Fernando Boavida, Edmundo Monteiro, and João Orvalho, editors, *Broadband Satellite Comunication Systems and the Challenges of Mobility*, Vol. 169, pp. 181–190. Springer-Verlag, New York, 2005. `http://link.springer.com/10.1007/0-387-24043-8_18`.

[56] Veth(4) - Linux manual page. `https://man7.org/linux/man-pages/man4/veth.4.html`. (Accessed at 2021-10-29).

[57] Pablo Neira-Ayuso, Rafael M. Gasca, and Laurent Lefevre. Communicating between the kernel and user-space in Linux using Netlink sockets. *Software: Practice and Experience*, pp. n/a–n/a, 2010. `http://doi.wiley.com/10.1002/spe.981`.

[58] P. Zheng and L.M. Ni. EMPOWER: A network emulator for wireline and wireless networks. Vol. 3, pp. 1933–1942. IEEE, 2003. `http://ieeexplore.ieee.org/document/1209215/`.

[59] Shie-Yuan Wang, Chih-Liang Chou, and Chun-Ming Yang. EstiNet openflow network simulator and emulator. *IEEE Communications Magazine*, Vol. 51, No. 9, pp. 110–117, September 2013. `http://ieeexplore.ieee.org/document/6588659/`.

[60] Takaaki Kawai, Shigeru Kaneda, Mineo Takai, and Hiroshi Mineno. A Virtual WLAN Device Model for High-Fidelity Wireless Network Emulation. *ACM Transactions on Modeling and Computer Simulation*, Vol. 27, No. 3, pp. 1–24, August 2017. `http://dl.acm.org/citation.cfm?doid=3130329.3067664`.

[61] Kunio Akashi, Tomoya Inoue, Shingo Yasuda, Yuuki Takano, and Yoichi Shinoda. NETorium: High-fidelity scalable wireless network emulator. pp. 25–32. ACM Press, 2016. http://dl.acm.org/citation.cfm?doid=3012695.3012699.

[62] Michael Dalton, David Schultz, Jacob Adriaens, Ahsan Arefin, Anshuman Gupta, Brian Fahs, Dima Rubinstein, Enrique Cauich Zermeno, Erik Rubow, James Alexander Docauer, Jesse Alpert, Jing Ai, Jon Olson, Kevin DeCabooter, Marc de Kruijf, Nan Hua, Nathan Lewis, Nikhil Kasinadhuni, Riccardo Crepaldi, Srinivas Krishnan, Subbaiah Venkata, Yossi Richter, Uday Naik, and Amin Vahdat. Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization. *the Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation*, pp. 373–387, 2018.

[63] Thomas Staub, Reto Gantenbein, and Torsten Braun. VirtualMesh: An emulation framework for wireless mesh networks in OMNeT++. ICST, 2009. http://eudl.eu/doi/10.4108/ICST.SIMUTOOLS2009.5563.

[64] Luca Veltri, Luca Davoli, Riccardo Pecori, Armando Vannucci, and Francesco Zanichelli. NEMO: A flexible and highly scalable network EMulatOr. *SoftwareX*, Vol. 10, p. 100248, July 2019. https://linkinghub.elsevier.com/retrieve/pii/S2352711019300135.

[65] Cloud Services - Amazon Web Services (AWS). https://aws.amazon.com/. (Accessed at 2021-10-29).

[66] Cloud Computing Services — Google Cloud. https://cloud.google.com/. (Accessed at 2021-10-29).

[67] Cloud Computing Services — Microsoft Azure. https://azure.microsoft.com/en-us/. (Accessed at 2021-10-29).

[68] Networking:bridge [Wiki]. https://wiki.linuxfoundation.org/networking/bridge. (Accessed at 2021-10-29).

[69] Open vSwitch. https://www.openvswitch.org/. (Accessed at 2021-10-29).

[70] Marco Antonio To, Marcos Cano, and Preng Biba. DOCKEMU – A Network Emulation Tool. pp. 593–598. IEEE, March 2015. http://ieeexplore.ieee.org/document/7096242/.

[71] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. Reproducible network experiments using container-based emulation. p. 253. ACM Press, 2012. http://dl.acm.org/citation.cfm?doid=2413176.2413206.

[72] Jiaqi Yan and Dong Jin. A lightweight container-based virtual time system for software-defined network emulation. *Journal of Simulation*, Vol. 11, No. 3, pp. 253–266, August 2017. https://www.tandfonline.com/doi/full/10.1057/s41273-016-0043-8.

[73] Home · adjacentlink/emane Wiki. https://github.com/adjacentlink/emane. (Accessed at 2021-10-29).

[74] Andreas Grau, Steffen Maier, Klaus Herrmann, and Kurt Rothermel. Time Jails: A Hybrid Approach to Scalable Network Emulation. In *2008 22nd Workshop on Principles of Advanced and Distributed Simulation*, pp. 7–14, Roma, Italy, June 2008. IEEE. http://ieeexplore.ieee.org/document/4545320/.

[75] Why Xen Project? https://xenproject.org/users/why-xen/.

[76] Rodney Martinez Alonso, David Plets, Yosvany Hervis Santana, Dariel Pereira Ruisanchez, Glauco Guillen Nieto, Luc Martens, and Wout Joseph. Emulation of a Dynamic Broadcasting Network with Adaptive Radiated Power in a Real Scenario. In *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–5, Valencia, Spain, June 2018. IEEE. https://ieeexplore.ieee.org/document/8436653/.

[77] Justin Yackoski, Babak Azimi-Sadjadi, Ali Namazi, Jason H. Li, Yalin Sagduyu, and Renato Levy. RFnestTM: Radio frequency network emulator simulator tool. In *2011 - MILCOM 2011 Military Communications Conference*, pp. 1882–1887, Baltimore, MD, USA, November 2011. IEEE. http://ieeexplore.ieee.org/document/6127587/.

[78] Hiroshi Mano and Shunsuke Saruwatari. Implementation and Evaluation of a Wireless System Emulator. *Journal of Information Processing*, Vol. 55, No. 3, pp. 1–14, 2014.

[79] Golsa Ghiaasi, Mehdi Ashury, Dimitrios Vlastaras, Markus Hofer, Zhinan Xu, and Thomas Zemen. Real-time vehicular channel emulator for future conformance tests of wireless ITS modems. In *2016 10th European Conference on Antennas and Propagation (EuCAP)*, pp. 1–5, Davos, Switzerland, April 2016. IEEE. http://ieeexplore.ieee.org/document/7481226/.

[80] Klaus Wehrle, Elias Weingärtner, and Hendrik vom Lehn. Device Driver-enabled Wireless Network Emulation. ACM, 2011. http://eudl.eu/doi/10.4108/icst.simutools.2011.245543.

[81] Gilles Silvano, Ivanovich Silva, Leonardo Oliveira, Marcos Pinheiro, and Bruno Ferreira. LVWNet: An hybrid simulation architecture for wireless sensor networks. *Design Automation for Embedded Systems*, Vol. 21, No. 3, pp. 139–155, December 2017. https://doi.org/10.1007/s10617-017-9191-y.

[82] What is Kubernetes? https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/. (Accessed at 2021-10-29).

[83] Ansible Documentation — Ansible Documentation. https://docs.ansible.com/ansible/latest/index.html. (Accessed at 2021-10-29).

[84] Platform Overview - Chef Progress. https://docs.chef.io/platform_overview/. (Accessed at 2021-10-29).

[85] Vagrant Introduction. https://www.vagrantup.com/intro. (Accessed at 2021-10-29).

[86] En:developers:documentation:cfg80211 [Linux Wireless]. https://wireless.wiki.kernel.org/en/developers/documentation/cfg80211. (Accessed at 2021-10-29).

[87] En:developers:documentation:mac80211 [Linux Wireless]. https://wireless.wiki.kernel.org/en/developers/documentation/mac80211. (Accessed at 2021-10-29).

[88] Michael Neufeld, Jeff Fifield, Christian Doerr, Anmol Sheth, and Dirk Grunwald. SoftMAC – Flexible Wireless Research Platform.

[89] ETSI. ETSI TS 102 687 v1.1.1 (2011-07) Intelligent Transport System (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 Ghz range; Access layer part, 2011.

[90] Hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator. `https://w1.fi/hostapd/`. (Accessed at 2021-10-29).

[91] Linux WPA Supplicant (IEEE 802.1X, WPA, WPA2, RSN, IEEE 802.11i). `https://w1.fi/wpa_supplicant/`. (Accessed at 2021-10-29).

[92] Unix(7) - Linux manual page, Unix - sockets for interprocess communication. `https://man7.org/linux/man-pages/man7/unix.7.html`. (Accessed at 2021-10-29).

[93] Mac80211 subsystem (basics) — The Linux Kernel documentation. `https://www.kernel.org/doc/html/v4.9/80211/mac80211.html`. (Accessed at 2021-10-29).

[94] Arata Kato, Mineo Takai, and Susumu Ishihara. Development of a Wireless LAN Emulation Framework based on Wireless Network Tap Device. *Journal of Information Processing*, Vol. 60, No. 1, pp. 27–37, 2019.

[95] Brendan Gregg. Flame Graphs visualize profiled code. `https://github.com/brendangregg/FlameGraph`, November 2021.

[96] ACPI Specification version 6.3, 2019.

[97] Intel Corporation. IA-PC HPET Specification. `https://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/software-developers-hpet-spec-1-0a.pdf`.

[98] World Bank. *Information and Communication Technology for Disaster Risk Management in Japan.* World Bank, Washington, DC, November 2019. `http://hdl.handle.net/10986/32797`.

[99] Benjamin Wisner, John Adams, and World Health Organization, editors. *Environmental Health in Emergencies and Disasters: A Practical Guide.* World Health Organization, Geneva, 2002.

[100] Masugi Inoue, Masaaki Ohnishi, Chao Peng, Ruidong Li, and Yasunori Owada. NerveNet: A Regional Platform Network for Context-Aware Services with Sensors and Actuators. *IEICE Transactions on Communications*, Vol. E94-B, No. 3, pp. 618–629, 2011. `http://joi.jlc.jst.go.jp/JST.JSTAGE/transcom/E94.B.618?from=CrossRef`.

[101] Hiroki Nishiyama, Masaya Ito, and Nei Kato. Relay-by-smartphone: Realizing multihop device-to-device communications. *IEEE Communications Magazine*, Vol. 52, No. 4, pp. 56–65, April 2014.

[102] Cabinet Office, Government of Japan. Quasi-Zenith Satellite System (QZSS). `https://www.q-anpi.qzss.go.jp/qzss/`. (Accessed at 2021-10-29).

[103] Cabinet Office, Government of Japan. QZSS Safety Confirmation Service (Q-ANPI). `https://qzss.go.jp/overview/services/sv09_q-anpi.html`. (Accessed at 2021-10-29).

[104] Association of Radio Industries and Businesses. ARIB Standards (ARIB STD-T98) Digital Convenience Radio Equipment for Simplified Service. `https://www.arib.or.jp/english/std_tr/telecommunications/desc/std-t98.html`, 2008.

[105] Hiroshi Mano, Hitoshi Morioka, and Tetsutaro Uehara. Experimental trial of Wireless LAN FILS (Fast Initial Link Setup). In *DICOMO2013*, pp. 1634–1639. IPSJ, 2013.

[106] Eng Hwee Ong. Performance analysis of fast initial link setup for IEEE 802.11ai WLANs. In *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, pp. 1279–1284, Sydney, Australia, September 2012. IEEE. `http://ieeexplore.ieee.org/document/6362543/`.

[107] Hiroki Kushida, Hiroshi Mano, Mineo Takai, Zhi Liu, and Susumu Ishihara. On the effectiveness of fils in IEEE 802.11ad wireless networks. In *2017 23rd Asia-*

Pacific Conference on Communications (APCC), pp. 1–6, Perth, WA, December 2017. IEEE. http://ieeexplore.ieee.org/document/8304047/.

[108] IEEE Computer Society. IEEE 802.11i-2004 - IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements.

[109] IEEE Computer Society. IEEE 802.1X-2020 - IEEE Standard for Local and Metropolitan Area Networks–Port-Based Network Access Control.

[110] John Vollbrecht, James D. Carlson, Larry Blunk, Bernard D. Aboba, and Henrik Levkowetz. Extensible Authentication Protocol (EAP). Request for Comments RFC 3748, Internet Engineering Task Force, June 2004. https://datatracker.ietf.org/doc/rfc3748.

[111] Daniel Simon, Ryan Hurst, and Bernard D. Aboba. The EAP-TLS Authentication Protocol. Request for Comments RFC 5216, Internet Engineering Task Force, March 2008. https://datatracker.ietf.org/doc/rfc5216.

[112] Ashwin Palekar, Simon Josefsson, Daniel Simon, and Glen Zorn. Protected EAP Protocol (PEAP) Version 2. Internet Draft draft-josefsson-pppext-eap-tls-eap-10, Internet Engineering Task Force, October 2004. https://datatracker.ietf.org/doc/draft-josefsson-pppext-eap-tls-eap-10.

[113] Wenchao Xu, Hassan Aboubakr Omar, Weihua Zhuang, and Xuemin Sherman Shen. Delay Analysis of In-Vehicle Internet Access Via On-Road WiFi Access Points. *IEEE Access*, Vol. 5, pp. 2736–2746, 2017.

[114] A. Mishra, Min Ho Shin, N.L. Petroni, T.C. Clancy, and W.A. Arbaugh. Proactive key distribution using neighbor graphs. *IEEE Wireless Communications*, Vol. 11, No. 1, pp. 26–36, February 2004. http://ieeexplore.ieee.org/document/1269714/.

[115] Junbeom Hur, Chanil Park, and Hyunsoo Yoon. An Efficient Pre-authentication Scheme for IEEE 802.11-Based Vehicular Networks. In Atsuko Miyaji, Hiroaki Kikuchi, and Kai Rannenberg, editors, *Advances in Information and Computer Security*, Vol. 4752, pp. 121–136. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. http://link.springer.com/10.1007/978-3-540-75651-4_9.

[116] IEEE Vehicular Technology Society. IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages - Amendment 1. Technical report, IEEE, 2017. https://ieeexplore.ieee.org/document/8065169/.

[117] ETSI TS 103 097 v1.3.1 - Intelligent Transport Systems (ITS); Security; Security header and certificate formats, 2017.

[118] Robil Daher and Alexey Vinel, editors. *Roadside Networks for Vehicular Communications: Architectures, Applications, and Test Fields.* IGI Global, 2013. http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-4666-2223-4.

[119] Michael Feiri, Rolf Pielage, Jonathan Petit, Nicola Zannone, and Frank Kargl. Pre-Distribution of Certificates for Pseudonymous Broadcast Authentication in VANET. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pp. 1–5, Glasgow, United Kingdom, May 2015. IEEE. http://ieeexplore.ieee.org/document/7146029/.

[120] Annette Böhm and Magnus Jonsson. Handover in IEEE 802.11p-based Delay-Sensitive Vehicle-to-Infrastructure Communication. *Research Report IDE - 0924*, 2007.

[121] Hiroki Nakano, Hitoshi Morioka, and Hiroshi Mano. IEEE 802.11-09/1000r3 - IEEE 802.11 for High Speed Mobility, 2009.

[122] Dmitry Bankov, Evgeny Khorov, Andrey Lyakhov, Ekaterina Stepanova, Le Tian, and Jeroen Famaey. What Is the Fastest Way to Connect Stations to a Wi-Fi HaLow Network? *Sensors*, Vol. 18, No. 9, p. 2744, September 2018. https://www.mdpi.com/1424-8220/18/9/2744.

[123] Wei Yin, Peizhao Hu, Wenbo Wang, Jiahui Wen, and Hongjian Zhou. FASUS: A fast association mechanism for 802.11ah networks. *Computer Networks*, Vol. 175, p. 107287, July 2020. https://www.sciencedirect.com/science/article/pii/S1389128619312083.

[124] Lyuye Zhang and Maode Ma. FKR: An efficient authentication scheme for IEEE 802.11ah networks. *Computers & Security*, Vol. 88, p. 101633, January 2020. https://www.sciencedirect.com/science/article/pii/S0167404818313373.

[125] Yuchung Cheng, Jerry Chu, Sivasankar Radhakrishnan, and Arvind Jain. TCP Fast Open. Request for Comments RFC 7413, Internet Engineering Task Force, December 2014. https://datatracker.ietf.org/doc/rfc7413.

[126] Nandita Dukkipati, Neal Cardwell, Yuchung Cheng, and Matt Mathis. Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses. Internet Draft draft-dukkipati-tcpm-tcp-loss-probe-01, Internet Engineering Task Force, February 2013. https://datatracker.ietf.org/doc/draft-dukkipati-tcpm-tcp-loss-probe-01.

[127] Ip(8) - Linux manual page. https://man7.org/linux/man-pages/man8/ip.8.html. (Accessed at 2021-10-29).

[128] En:users:documentation:iw [Linux Wireless]. https://wireless.wiki.kernel.org/en/users/documentation/iw. (Accessed at 2021-10-29).

[129] Christoph Sommer and Falko Dressler. *Vehicular Networking*. Cambridge University Press, December 2014.

[130] Kenneth Laberteaux and Hannes Hartenstein. *VANET: Vehicular Applications and Inter-Networking Technologies*. John Wiley & Sons, November 2009.

[131] FCC Allocates Spectrum 5.9 GHz Range for Intelligent Transportation Systems Uses. https://transition.fcc.gov/Bureaus/Engineering_Technology/News_Releases/1999/nret9006.html. (Accessed at 2021-10-29).