Emergency Message Transmission Method Based on Vehicular Ad Hoc Networks (Postprint)

Authors: Mu Jian, high strength, YIN Kexin, Zhu Jianqi

Date: 2018-05-02T00:00:00+00:00

Abstract

This study investigates the information transmission delay problem in vehicular ad-hoc networks and proposes a transmission method for emergency messages. By intercepting packet data information through the Netfilter framework and reassembling safety warning information packets using Linux virtual devices, the method directs them directly to the physical network card for transmission to the destination terminal. The transmission process reduces traditional buffer waiting time and additional overhead from TCP/IP peer layers; results demonstrate that it effectively reduces transmission delay and improves vehicle driving safety.

Full Text

Emergency Message Transmission Method Based on VANET

Mu Jian[1], Gao Qiang[1], Yin Kexin[2], Zhu Jianqi[1]† (1. College of Computer Science and Technology, Jilin University, Changchun 130012, China; 2. College of Computer Science and Engineering, Changchun University of Technology, Changchun 130012, China)

Abstract: This paper investigates the issue of information transmission delay in Vehicular Ad-hoc Networks (VANET) and proposes an emergency message transmission method. The method captures packet data information through the Netfilter framework, reassembles safety warning messages using Linux virtual devices, and directs them to the physical network card for transmission to the destination terminal. This transmission process reduces both the traditional cache waiting time and the additional overhead of TCP/IP peer layers. Results demonstrate that the method effectively reduces transmission delay and improves vehicle driving safety.

Keywords: VANET; emergency message; Netfilter architecture; virtual device

0 Introduction

In simulated safety accident experiments, lower system response time enables safety warning information to reach drivers earlier, effectively providing them with more reaction time. This plays a crucial role in preventing accidents. Vehicular Ad-hoc Network (VANET) [1,2] is a distributed, short-range communication network dynamically constructed between vehicles and between vehicles and roadside infrastructure on roadways. Its actual network structure is shown in Figure 1 [Figure 1: see original paper]. VANET has given rise to numerous V2X modes, including Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Network (V2N), and Vehicle-to-Pedestrian (V2P). Typical applications of VANET technology encompass driving safety warnings, assisted driving, distributed traffic information dissemination, and traffic flow control. A primary objective of VANET construction is to improve vehicle driving safety by enabling timely sharing of safety-related information among vehicles, which imposes high timeliness requirements on the transmission and reception of safety-critical emergency messages.

In recent years, substantial research has been conducted on VANET protocol stacks [3,4] and routing algorithms [5~8]. To achieve efficient data routing transmission, reference [10] proposed a data transmission strategy based on parked vehicle backbone networks, which forms a virtual network using parked vehicles for data transmission. This approach achieves higher transmission success rates and lower transmission delays compared to traditional protocols, though the success rate decreases as vehicle message generation rates increase. Additionally, packet size and RSU quantity affect protocol performance. Reference [11] proposed a distance-based receiver-oriented routing protocol (BDRO) for emergency messages, featuring selectivity, passivity, and forwarding node candidate mechanisms that use distance from the source node as the basis for selecting forwarding nodes. However, emergency message transmission still employs the TCP/IP protocol method, which could be further simplified for emergency scenarios. Reference [12] introduced a VANET collaborative safety warning routing algorithm for highway safety requirements, utilizing directional broadcast forwarding to achieve rapid propagation. While its approach in packet generation and reception phases is similar to normal packet transmission, existing message sending methods primarily rely on the traditional TCP/IP protocol approach.

The TCP/IP protocol adopts a layered modular protocol stack design, mainly divided into the application layer, transport layer, network layer, and network interface layer. Each layer completes a portion of data packet transmission functionality, with packets being packaged sequentially from top to bottom through the protocol stack. However, in VANET applications for emergency message transmission, TCP/IP exhibits several deficiencies. First, before the transport layer and IP layer place application layer packets into the network interface layer for transmission, they require hierarchical packaging (adding headers), which may involve data parsing and copying. For emergency messages, whether data parsing, copying, or the mandatory packaging at the transport and IP layers,

all increase transmission delay. Second, regarding the IP layer, its primary function is adding IP headers and finding appropriate routes through different IP addresses to send packets to designated terminals. In VANET, safety-related emergency messages only need to be transmitted to vehicles within a certain range around the accident vehicle, making broadcast the most suitable approach. Moreover, vehicular networks employ the 802.11p wireless communication protocol with an effective transmission range of 300 meters, which is sufficient for emergency message transmission scope. From this perspective, TCP/IP' s IP layer functionality is unnecessary for emergency message transmission and only increases delay. Third, regarding the transport layer, its main function is adding transport layer headers and implementing connection control (reliable transmission, error control, flow control). In VANET, safety-related emergency messages require minimal application processing, and accident vehicles continuously broadcast emergency messages until the accident is resolved. Therefore, transport layer functionality is unnecessary for emergency message transmission and actually increases delay. Consequently, when transmitting safety-related emergency messages in VANET, traditional TCP/IP' s transport layer and IP layer functions are not only unnecessary but also increase transmission delay.

Based on this analysis, we propose the protocol model shown in Figure 2 [Figure 2: see original paper]. The model introduces an emergency message processing module whose transmission process reduces traditional cache waiting delays and eliminates the overhead of unnecessary IP and transport layer functions in TCP/IP, preempting other transmission resources to directly access the physical layer. This achieves the goal of immediately sending emergency messages upon generation.

According to the protocol model in Figure 2, this paper presents an emergency message transmission method that captures packet data information through the netfilter architecture, reassembles safety warning messages using Linux virtual devices, and directly points them to the physical network card for transmission to the destination terminal. The flow diagram of this method is shown in Figure 3 [Figure 3: see original paper], with the following main steps:

a) Emergency message generation phase: When an emergency occurs, the emergency message processing module in the vehicle generates emergency messages. The emergency message format is shown in Figure 4 [Figure 4: see original paper]. Upon accident or other emergencies, the system process of the emergency message processing module in the vehicle' s onboard system preempts CPU resources to quickly generate corresponding emergency messages, or authorized user processes call the emergency message processing module to generate messages.

b) Emergency message sending phase: The emergency message processing module organizes and sends the generated emergency messages. It adds necessary location information and emergency message types (e.g., vehicle collision, ambulance, collaborative collision avoidance) and ensures the emergency message length is within 20 bytes, as IP packets have a mini-

mum effective length of 20 bytes. This allows both the emergency message processing module and TCP/IP network layer to distinguish message types by length. The module then passes the organized emergency message directly to the network interface layer for broadcast transmission within the 300-meter effective range of the 802.11p wireless communication protocol.

c) Emergency message reception phase: The emergency message processing module receives and processes emergency messages. Based on message length information, the module determines whether the received message is an emergency message. When surrounding vehicles' network interface layers receive packets, they pass them to both the TCP/IP network layer and the emergency message processing module. The TCP/IP network layer directly discards packets shorter than 20 bytes, while the emergency message processing module discards packets longer than 20 bytes and processes emergency messages shorter than 20 bytes. Upon receiving an emergency message, the corresponding system process preempts CPU resources to prioritize processing safety-related emergency messages.

As shown in Figure 4, the message type field in the emergency message format occupies 16 bits (2 bytes), distinguishing up to 65,535 different types of emergency events (post-accident warning, vehicle collaborative collision avoidance, intersection collaborative collision avoidance, etc.). Reserved bits are for potential future extensions, and the data portion (maximum 15 bytes) primarily stores vehicle location information. The maximum emergency message length is 19 bytes, while IP packets have a minimum effective length of 20 bytes, enabling message type differentiation by length.

1 Emergency Message Transmission Method Characteristics

The proposed emergency message transmission method for VANET has the following characteristics:

a) It bypasses the packaging processes of TCP/IP' s transport and IP layers when transmitting emergency messages in VANET.

b) It employs a dedicated system kernel module to process emergency messages, enabling convenient CPU preemption and eliminating process queuing time.

Using Changchun as an example, when a collision occurs at Satellite Square, the emergency message generation unit of the emergency message processing module in the accident vehicle' s onboard system, triggered by external hardware, calls the system process to preempt CPU resources and quickly generate corresponding emergency messages. Alternatively, authorized user processes can call this system process to generate emergency messages.

The emergency message sending unit adds the accident vehicle' s location information "Satellite Square" to the emergency message' s "data" field, occupying

8 bytes. Assuming "0000000000000001" represents a collision accident type, this value is added to the "message type" field, occupying 2 bytes. If no other important information exists, the "reserved bits" field remains empty, making the emergency message 10 bytes long. The organized emergency message is then passed to the network interface layer for broadcast transmission within the 300-meter effective range of the 802.11p wireless communication protocol.

When surrounding vehicles' network interface layers receive this emergency message, they pass it to both the TCP/IP network layer and the emergency message processing module' s receiving unit. The TCP/IP network layer directly discards the emergency message because its length is less than 20 bytes, while the emergency message processing module' s receiving unit identifies it as an emergency message based on its length and calls the system process to preempt CPU resources for prioritized processing.

2 Implementation of Emergency Message Sending Mechanism

Figure 5 [Figure 5: see original paper] shows the structure of the VANET emergency message sending mechanism implementation. We established a local area network test environment using two normally networked PCs as the development platform (sender) and test platform (receiver), with linux-gcc-3.2.2-5 as the overall compilation environment and Red Hat 9.0 Linux as the operating system. All PCs in the LAN simultaneously support IPv4/IPv6 address protocols.

The implementation primarily completes the following tasks: user space runs IPv6-based TCP packet sending processes (tcpclient), TCP packet receiving processes (tcpserver), and UDP packet receiving processes (udpserver). In the Linux kernel space, two important modules are loaded and operate normally: the packet module containing netfilter architecture and the vdev module containing Linux virtual devices, both derived from Linux device drivers. These programs combine to implement VANET emergency message generation and immediate transmission in Linux-based vehicular high-speed mobile terminals.

2.1 Netfilter Architecture

Emerging in Linux kernel 2.4, netfilter is a firewall architecture that primarily implements state detection and packet filtering functions. Its open interface allows implementing custom functional modules at corresponding protocol layers. The netfilter architecture defines five hook functions for both IPv4 and IPv6 protocols, as shown in Figure 6 [Figure 6: see original paper]. For IPv6, these are:

HOOK(1): NF_IP6_PRE_ROUTING
HOOK(2): NF_IP6_LOCAL_IN
HOOK(3): NF_IP6_FORWARD

HOOK(4): NF_IP6_POST_ROUTING
HOOK(5): NF_IP6_LOCAL_OUT

The VANET emergency message sending mechanism' s packet module primarily uses the NF_IP6_LOCAL_OUT hook function among netfilter' s five hooks. This captures packets when sent from local processes for connection tracking and packet processing. The specific data processing workflow is: when various data packets in the network enter the operating system from the data link layer, they first undergo IP address verification, where HOOK(1) (NF_IP6_PRE_ROUTING) determines whether the packet enters the local machine or is forwarded. Forwarded packets are caught by HOOK(3) (NF_IP6_FORWARD); packets entering the local machine are caught by HOOK(2) (NF_IP6_LOCAL_IN) and passed upward. Forwarded packets are caught by HOOK(4) (NF_IP6_POST_ROUTING) after forwarding. Packets processed by the local machine are caught by HOOK(5) (NF_IP6_LOCAL_OUT), then after routing processing, caught again by HOOK(4) (NF_IP6_POST_ROUTING) before being transmitted back to the network. Each hook function returns an integer constant after processing, and the Linux kernel' s corresponding handling of the packet depends on these return values:

a) NF_DROP: Discard without processing

b) NF_ACCEPT: Accept for next processing step

c) NF_STOLEN: Anomalous packet

d) NF_REPEAT: Re-enter this hook

e) NF_QUEUE: Enter user space queue for corresponding process handling

2.2 Linux Device Driver Technology

In Linux, everything is a file, with actual hardware corresponding to "device files."These devices mainly fall into three types: character devices, block devices, and network interfaces, existing independently of the kernel and capable of being inserted when necessary and removed when not. The VANET emergency message sending mechanism primarily applies network device driver technology, i.e., network interfaces.

The Linux device driver structure is shown in Figure 9 [Figure 9: see original paper]. The "vdev" module used in this paper implements three functions: obtaining useful information from TCP packets in the "packet" module containing netfilter architecture; extracting the original information to be sent and reassembling it into compliant UDP emergency messages; and directing the newly assembled UDP packets straight to network card "eth0" for transmission without passing through the protocol stack or various cache queues.

The VANET emergency message sending mechanism combines Linux socket network programming, netfilter architecture, and Linux device driver technology. The detailed workflow is shown in Figure 10 [Figure 10: see original paper]. The sending mechanism can be divided into three parts based on where programs run in the computer architecture: first, process programs running in user space; second, the Linux operating system managing user and kernel space and the network protocol stack; third, two kernel modules closely related to the accelerated packet sending method—the packet module and vdev module. Based on the entire network communication process, it can also be divided into two parts: the sending terminal responsible for generating and sending VANET emergency messages, and the receiving terminal responsible for receiving emergency messages and verifying the mechanism' s normal operation.

The first part operates at the highest layer of the network protocol, primarily completing user requirements. It involves both the sending terminal for generating and sending emergency messages and the receiving terminal for receiving emergency messages and testing the mechanism. Among the three key technologies, it uses Linux socket network programming [9] to implement the overall network information transmission process, specifically divided into three independent processes: the TCP packet sending process (tcpclient) running on the sending terminal to generate and send TCP emergency messages, the TCP packet receiving process (tcpserver) running on the receiving terminal to receive TCP emergency messages captured but unmodified by netfilter architecture, and the UDP packet receiving process (udpserver) running on the receiving terminal to receive UDP emergency messages that are captured and processed by netfilter architecture, reassembled in the virtual device, and finally sent directly to the network card (eth0).

The second part is the Linux operating system managing user and kernel space and the network protocol stack related to network information transmission. This remains unmodified in the proposed mechanism, still using the TCP/IP protocol stack.

The third part consists of two kernel modules running in kernel space closely related to the accelerated packet sending method: the packet module containing Linux netfilter architecture technology and the vdev module implementing virtual device technology. In the packet module, the primary technology is Linux netfilter architecture. First, the packet module is loaded into the Linux kernel using the insmod command. Then, the NF_IP6_LOCAL_OUT hook function provided by netfilter captures relevant TCP packets, stores them in the sk_buff structure, obtains the data information to be transmitted using pointer movement, and passes this data to the vdev module. Simultaneously, the netfilter architecture returns NF_ACCEPT, meaning the captured TCP emergency messages return to the capture point and continue transmission according to the traditional sending mechanism, indicating that the sending terminal sends two types of emergency messages with the same information in TCP and UDP formats. After all tasks are completed, the rmmod command unloads the packet

module. Thus, the packet module completes its two main functions: obtaining packet information and passing emergency information to the vdev module.

The vdev module must first be loaded into the kernel using the insmod command, with lsmod verifying successful loading. The packet module calls the vdev_module_rx(data) function to pass useful information from TCP emergency messages to the vdev module. The vdev_module_rx function converts the obtained useful information into UDP emergency messages, then calls the vdev_xmit function within vdev_module_rx to send the reassembled UDP packets directly to the network card (eth0). After task completion, the vdev module can be unloaded using the rmmod command. Through this process, the vdev module completes its three main functions: obtaining useful information from the packet module, reassembling this information into UDP emergency messages, and directly sending them to the network card (eth0).

We implemented a VANET emergency message sending method test program operating within a LAN range, with IPv6 protocol stacks configured on all communicating hosts. The sending terminal runs only one TCP packet sending process (tcpclient) that generates and sends emergency messages related to driving safety stored in text files. The receiving terminal simultaneously runs a TCP packet receiving process (tcpserver) for receiving TCP emergency messages and a UDP packet receiving process (udpserver) for receiving UDP emergency messages, both receiving packets containing the same emergency information. Test results show that both TCP and UDP emergency message receiving processes successfully receive emergency message information, demonstrating that the researched and implemented VANET emergency message sending mechanism accurately completes emergency message transmission. Additionally, actual operation reveals that the UDP emergency message receiving process receives packets faster than the TCP process, effectively reducing emergency message transmission delay.

3 Conclusion

VANET emergency message sending mechanism is a novel research field involving communication technology, network technology, computer technology, and transportation technology. This paper thoroughly investigates and analyzes packet sending mechanisms and proposes a VANET emergency message transmission method. The method captures packet data information through netfilter architecture hooks, reassembles safety warning messages using Linux virtual devices, and directly sends them to the destination receiving terminal via the physical network card. When transmitting emergency messages in VANET, this method bypasses the packaging processes of TCP/IP' s transport and IP layers while using dedicated system kernel modules to process emergency messages, enabling convenient CPU preemption and eliminating process queuing time.

Compared with routing forwarding protocols [12], the emergency message transmission mechanism enables the emergency message processing module to quickly

generate corresponding emergency messages when vehicles encounter accidents or other emergencies, or allows authorized user processes to call the emergency message processing module to generate emergency messages. During the sending phase, emergency messages are controlled within 20 bytes and delivered to the network interface layer for broadcast. During the receiving phase, both the TCP/IP network layer and emergency message processing module receive messages, with the latter preempting CPU resources to prioritize processing safety-related emergency messages upon receipt.

The main contributions of this paper are:

a) Developed a relatively complete solution and implementation for the VANET emergency message sending mechanism.

b) Selected appropriate software and hardware environments for mechanism development, including Red Hat 9.0 Linux as the development platform operating system and a pure IPv6 test environment.

c) Completed high-level application development for the VANET emergency message sending mechanism using socket network programming technology.

d) Designed an emergency message format that effectively distinguishes emergency messages from ordinary packets, preparing for the proposed VANET emergency message sending mechanism.

e) Implemented emergency message information capture in the VANET emergency message sending mechanism using Linux netfilter architecture technology.

f) Applied netfilter architecture technology to process emergency message information, reassemble it into UDP emergency messages, and send them directly to the network card for rapid forwarding.

References

[1] Rene O, Carlos M, Azzedine B, et al. Reliable datadissemination protocol for VANET traffic safety applications [J]. Ad hoc Networks, 2017, 63: 30-44.
[2] Jun Tao, Yifan Xu, Ziyi Zhang, et al. A resource allocation game with restriction mechanism in VANET cloud [J]. Concurrency & Computation Practice & Experience, 2016, 29 (14).
[3] Barakat P M, Tarek R S, Khaled S. Performance study of manet routing protocols in VANET [J]. Arabian Journal for Science & Engineering, 2016.
[4] Tariq E, Layth A K A D, Yamaan M. Review and performance comparison of VANET protocols: AODV, DSR, OLSR, DYMO, DSDV & ZRP [C]// Proc of AIC-MITCSA. 2016: 1-6.
[5] Raj K J, Jaidhar C D. Location prediction algorithm for a nonlinear vehicu-

lar movement in VANET using extended Kalman filter [J]. Wireless Networks, 2016: 1-16.
[6] Lochert C, Hartenstein H, Tian J, et al. A routing strategy for vehicular ad hoc networks in city environments [C]// Proc of IEEE Intelligent Vehicles Symposium. 2003, 156-161.
[7] Sharma H L, Agrawal P, Kshirsagar R V. Multipath reliable range node selection distance vector routing for VANETT: design approach [C]// Proc of International Conference on Electronic Systems, Signal Processing and Computing Technologies. 2014: 280-283.
[8] Feng Huifang, Liu Chunfeng, Shu Yantai, et al. Location Prediction of Vehicles in VANET Using A Kalman Filter [J]. Wireless Personal Communications, 2015, 80 (2): 543-559.
[9] 杨秋黎, 金智. Windows 网络编程 [M]. 2 版. 北京: 人民邮电出版社, 2016.
[10] 朱金奇, 马春梅, 刘明, 等. 车载自组织网络中基于停车骨干网络的数据传输 [J]. 软件学报, 2016, 27 (2): 432-450.
[11] 冯硕. VANET 中紧急报文路由机制研究 [D]. 长春: 吉林大学, 2014.
[12] 徐波, 徐守志, 郭鹏飞, 等. 基于最短时延优先的交通安全预警 VANET 路由协议 [J]. 华中科技大学学报: 自然科学版. 2013, 41 (S2).

Note: Figure translations are in progress. See original paper for figures.