Hindawi
Mobile Information Systems
Volume 2017, Article ID 7479267, 9 pages
https://doi.org/10.1155/2017/7479267

*Research Article*

# Network Intelligence Based on Network State Information for Connected Vehicles Utilizing Fog Computing

**Seongjin Park and Younghwan Yoo**

*Department of Electrical and Computer Engineering, Pusan National University, Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan 46241, Republic of Korea*

Correspondence should be addressed to Younghwan Yoo; ymomo@pusan.ac.kr

Received 9 December 2016; Accepted 30 January 2017; Published 20 February 2017

Academic Editor: Yujin Lim



This paper proposes a method to take advantage of fog computing and SDN in the connected vehicle environment, where communication channels are unstable and the topology changes frequently. A controller knows the current state of the network by maintaining the most recent network topology. Of all the information collected by the controller in the mobile environment, node mobility information is particularly important. Thus, we divide nodes into three classes according to their mobility types and use their related attributes to efficiently manage the mobile connections. Our approach utilizes mobility information to reduce control message overhead by adjusting the period of beacon messages and to support efficient failure recovery. One is to recover the connection failures using only mobility information, and the other is to suggest a real-time scheduling algorithm to recover the services for the vehicles that lost connection in the case of a fog server failure. A real-time scheduling method is first described and then evaluated. The results show that our scheme is effective in the connected vehicle environment. We then demonstrate the reduction of control overhead and the connection recovery by using a network simulator. The simulation results show that control message overhead and failure recovery time are decreased by approximately 55% and 5%, respectively.

## 1. Introduction

New IT technologies have recently been converged for a new platform, due to the prevalence of smartphones, formidable growth in mobile traffic, and the emergence of IoT services. Recent research has several common themes. First, it considers large numbers of wireless mobile devices, of many types (e.g., portable smart devices, cars, drones, and mobile sensors). Second, besides data collection in the heterogeneous communication environment, it emphasizes intelligent management and utilization of data in accordance with specific services. This work encompasses new technologies including fog computing, Software Defined Networks (SDN), and Fifth-Generation (5G) networks.

The term "fog computing" was coined by Cisco Systems in 2012. Cloud computing is a technology where users can take IT resources (a remote server, communication systems, storage, services, etc.) anywhere. However, cloud computing cannot provide real-time services, which require less than 1 ms delay of a 5G requirement because of the physical distance between users and a remote server. On the other hand, fog computing can provide more practical 5G services, by deploying servers nearer to the end user. The fog metaphor comes from the idea that it utilizes small "close to the ground" servers, as opposed to more distant servers in cloud computing [1]. In fog computing, cloud servers are located near the end user.

Fog computing adds a fog layer to the cloud computing between the cloud server and the end user. Access points, base stations, routers, and mobile devices can serve as fog servers. Mobile devices can overcome their lack of adequate IT resources by using fog computing. Besides, it is possible to take advantage of mobility pattern of mobile devices. Examples of applications and services that can utilize fog computing include connected vehicles (CV), smart grid, wireless sensor and actuator networks (WSAN), augmented reality (AR), content delivery networks (CDN), and others that require real-time and big data [2]. Communication

between a fog server and a mobile device can use various wireless technologies (WiFi, Bluetooth, LTE, and 5G) depending on the service type.

Efficient resource management is essential but difficult, because of the inflexibility of the current network structure. One switch or router has to perform all functions, including forwarding, routing, and management of network resources. Each switch or router performs these tasks consecutively. These hardware-dependent features cause inefficient resource utilization with time-varying network state. In order to modify some network functions, it takes too much time to correct all the switch or router which is incompatible with each vendor-specific device. In addition, this can result in temporary outages.

SDN was introduced in order to overcome these drawbacks of the current network architecture [3]. It represents a paradigm shift from hardware-dependent to software defined architectures. Its main benefit is to decouple the control plane from the data plane. In the past, the global Internet had to be realized as a distributed system. High data rates in the backbone network now make it possible to centralize the networks control system. Thus, a central controller can manage the overall network state and install forwarding rules in each network device. To achieve this, it is necessary to separate hardware from software functions in vendor-specific devices. Network devices simply forward incoming packets according to a flow table installed by the controller. Other functions, such as routing path calculation and global resource management, are performed by the controller. In this respect, fog computing applies the SDN architecture as a network perspective.

The SDN is one of the indispensable components of the 5G network, which has to handle high volumes of mobile traffic and support low latency communication. Because one technology cannot satisfy all user demands, it is important for emerging technologies to cooperate with existing infrastructure for seamless services. From this perspective, the centralized resource management of SDN brings about efficiency benefits in multiservice and multitechnology environments [4]. Then, the fog computing may enhance the benefits of SDN since the fog layer reduces the distance between mobile devices and servers, resulting in the improvement of communication latency.

In this paper, we apply our ideas to the field of connected vehicles which has attracted much attention these days. Cars are now smart devices, with numerous sensors and communication capabilities. Vehicles are less energy-constrained and can have higher communication capacity than smartphones. Moreover, there are many challenges to find new applications concerning the devices mobility. The connected vehicle is an emerging representative application. A vehicle is not only a mobile node but also a forwarding device which can act as infrastructure, like a base station or a RSU (Roadside Unit) [5].

Services in connected vehicles are classified as being safety-related or non-safety-related. A safety-related service is critical for safe driving. A non-safety-related service is not critical but requires a high data rate to ensure a seamless user experience. For example, a self-driving car must promptly and accurately collect and analyze data from its own embedded sensors, from neighboring cars, and from infrastructure, in order to maintain safety. In this paper, we consider the connected vehicle as an application that takes advantage of fog computing and SDN. However, the proposed ideas are not limited to only the connected vehicle.

In this paper, we categorize mobility information into three classes and measure the signal strength of the link between the controller and each switch in order to help the controller supervise global network resources. We further propose an intelligent maintenance method utilizing network information and give some practical use cases for connected vehicles.

## 2. Related Work

*2.1. Mobile SDN.* First, we introduce recent research related to SDN in the mobile environment. Software Defined Wireless Network (SDWN) [6] is the first SDN study using the IEEE 802.15.4 (Low Rate Wireless Personal Area Networks) standard. This architecture decouples a sink node from a general node. This work consider only wireless channel but not node mobility.

Software Defined Mobile Networks (SDMN) [7] consider the mobile environment in the Radio Access Network (RAN) and the core network. Ku et al. [8] applied SDN to Mobile Ad Hoc Networks (MANETs) and Vehicular Ad Hoc Networks (VANETs). The major difference from the precedent researches is the wireless communication between the controller and switches. In addition, moving nodes add complexity. The LTE link is used for control plane communications, and the WiFi link is used by the data plane. An SDN wireless node simultaneously plays the roles of a router and a host. To obtain the current topology, each SDN wireless node sends periodic beacon messages. The nodes respond with connection information, which the controller maintains in a table. Because of the nature of wireless communications, each switch must prepare for link failure between the controller and switches. In the general case, SDN switches forward incoming packets to a corresponding output port in accordance with an installed flow table; similarly, an SDN wireless node forwards packets to an appropriate wireless interface or channel in MANET. In other words, the role of the port in the SDN switch is replaced by the wireless interface or the frequency in the SDN wireless node.

*2.2. Fog Computing.* Fog SDN (FSDN) [9] is an architecture that combines fog computing and SDN in VANETs. In a VANET, mobile nodes move faster than those in a MANET, and their mobility pattern tends to follow fixed roadways. In addition to the SDN controller and the SDN wireless node used in an SDN-based MANET, an FSDN includes an SDN Roadside Unit (RSU), an SDN Roadside Unit Controller (RSUC), and a base station as a fog device. All fog devices are controlled by the SDN controller. The SDN controller knows the global state of the network and assigns tasks at a service level to efficiently utilize limited resources. Acting as a fog device, an SDN RSU, an SDN RSUC, or a base station can supply low latency services to mobile devices, by

using attributes of each device location and mobility pattern. Examples are given for a safety-related lane-change service and a non-safety-related data streaming service.

Vehicular Fog Computing (VFC) [5] is an architecture which treats vehicles as infrastructure. The main difference between VFC and Vehicular Cloud Computing (VCC) is that the former can support real-time services and geolocation-based distributed services, due to the proximity of mobile users to fog servers. Using vehicles as infrastructure in VFC avoids the additional cost of dedicated infrastructure. This work distinguishes between moving cars and parking cars in the urban environment and divides computational and communication resources in accordance with their usage. For example, rapidly moving cars are preferred for relaying data to a distant location, while slowly moving cars, passing through a congested area, are favorable as localized computational resources. Parked cars are used as infrastructure (e.g., RSU, RSUC, or base station).

Previous research has proposed architectures applying SDN and fog computing to MANET or VANET. However, these works do not sufficiently utilize mobility information. SDN wireless nodes simply inform the fog server or the SDN controller periodically about their current connection to neighboring nodes. We propose a method of managing and utilizing mobility information in the connected vehicle.

## 3. The Proposed Idea

The concept of SDN can facilitate easy and flexible management of a network. It is essential in the network layer of the fog computing. The SDN architecture consists of three layers: application layer, network controller layer, and infrastructure layer. The infrastructure layer is responsible for packet forwarding; and the network controller layer collects network state information and provides the API to control forwarding devices in the infrastructure layer. Finally, the application layer decides the forwarding policy using network intelligence. The layers communicate with each other through open interfaces. A northbound API provides the interface between application layer and network controller layer. A southbound API is utilized for the communication between network controller and infrastructure layer. OpenFlow [10] is a common example of a southbound API. Several northbound APIs are being standardized.

Then, there are some challenges to run an SDN controller on fog servers. Since the fog servers maintain mobile nodes as the clients, the SDN controller should also consider the nodes mobility information and the vulnerable nature of communications in the mobile environment. To do this, we propose a management method of the network state information at first and then suggest some use cases.

*3.1. Management of Network State Information.* The controller maintains network state information (NSI) to manage the network. Node mobility information is especially important in unpredictable environments, such as connected vehicles. Previous research [8] only utilizes the nodes connection state (i.e., up or down) in a mobility table. We augment this with the Received Signal Strength Indicator (RSSI) to measure the channel quality. It is also desirable to exploit node mobility patterns. This paper identifies three types of mobility pattern: nearly stationary nodes, nodes that move in arbitrary patterns, and nodes which move in predictable patterns. By categorizing mobility patterns, the controller can coordinate the overall network from a global perspective.

Figure 1 shows an example of the system in a campus scenario. A school bus is an example of a moving node which has a predictable mobility pattern. Similarly, a moving car and a parked car represent an unpredictable moving node and a quasi-stationary node, respectively. The controller manages the parameters shown in the following part.

*Structure of NSI*

$V$: Set of vehicles, $V = \{v_1, v_2, \ldots, v_{|V|}\}$

$N_l$: A list of neighbor nodes, $N_l \subset V$

$N_p$: The current position of the node, $N_p = (x, y, z)$

$N_m$: A class of the node's mobility, $N_m = \{r, s, p\}$

$N_r$: RSSI information of the node, $N_{r,\text{th}}$: threshold value

$N_f$: Failure type of the node, $N_f = \{t, s\}$

$N_d$: Distance from the controller, $N_c$: max distance

The parameter $N_m$ denotes the mobility class of vehicles, where the value $r$ indicates a vehicle which has random movement, $s$ is a quasi-stationary vehicle, and $p$ is a vehicle which moves in a predictable pattern.

*3.2. Adaptive Control Messages.* Mobility information is used to reduce control message overhead. Each wireless node sends and receives beacon messages periodically, in order to maintain recent connection information. In mobile networks with dynamic topologies, the number of control messages increases with the number of nodes. This scalability problem can be addressed by leveraging node mobility patterns. If nodes mobility pattern is quasi-stationary or predictable, we can reduce the reporting frequency, because the controller can predict the nodes position. On the other hand, nodes that have an unpredictable mobility pattern must report their mobility information frequently in order to maintain the current network topology.

*3.3. Connection Recovery Process.* NSI is also useful for recovery. First, we consider connection failure between the fog server and the wireless nodes. Since the link between the nodes and the connected vehicle is wireless, the networks suffer from frequent disconnection, which diminishes overall network performance. Prior research [8] suggests recovery by reverting to routing policies such as Ad Hoc On-Demand Distance Vector Routing (AODV), or Dynamic Source Routing (DSR). Although this is a very easy and simple solution, it is not sufficient for the intelligent network. Due to the unstable communication environment in the connected vehicle, networks must predict and cope with the communication interruption. We propose that the controller predict whether the connection between a fog server and a

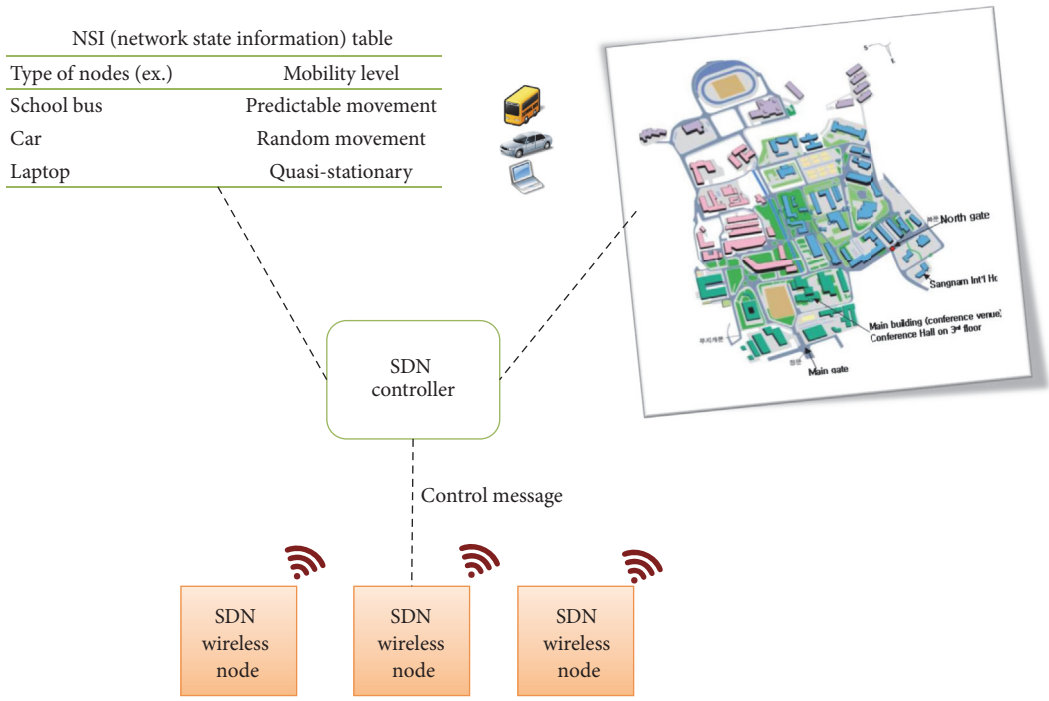



Figure 1: Overall system overview.

**Begin**
**while** $N_r \leq N_{r,\text{th}}$ **do**
  **if** $N_d \leq N_c$ **then**
    $N_f == t$ ▷ Temporal Failure
  **else**
    **if** $N_m == r$ **then**
      $N_f == s$ ▷ Severe Failure
    **else if** $N_m == s \parallel N_m == p$ **then**
      Predicts the node's upcoming position
      **if** $N_d \leq N_c$ **then**
        $N_f == t$ ▷ Temporal Failure
      **else**
        $N_f == s$ ▷ Severe Failure
      **end if**
    **end if**
  **end if**
**end while**
**End**

Algorithm 1: Failure decision process.

wireless node will be lost, by using the nodes mobility pattern and the link quality. The controller also decides whether a disconnected nodes link will recover soon or not. Algorithm 1 shows the controllers decision process. When the controller forecasts a connection loss, it is classified as either a temporal or a severe failure. The recovery process then proceeds as shown in Algorithm 2. In a predicted temporal failure, the disconnected node simply waits until the link recovers. If a severe connection loss is predicted, the node ceases to follow the existing forwarding table and performs its own routing policy in advance of the disruption. For example, when the controller predicts that a lost connection node will undergo a failure for a short time, returning to the existing routing protocol is inefficient from the perspective of the network resource utilization.

*3.4. Server Failure Recovery Process.* We further propose a recovery process from fog server failure. The vehicle that was connected to the failed fog server cannot continue to use its services until the connection is resumed. To offer

```
Begin
if N_f == t then
    Stay until recover the connection
else if N_f == s then
    Delete the current flow table & Run an existing routing protocol
end if
End
```

Algorithm 2: Recovery process.

seamless services, a fast fail-over scheme is needed. The fast fail-over scheme can be regarded as an optimization problem. Its objective is to find the maximum number of services that can be recovered while meeting the delay requirement of each service. We define this problem as a real-time scheduling optimization problem in Section 4. Then, we suggest our scheduling algorithm for the mobile environment like Figure 1. The simulation result of the control overhead and the connection recovery process is shown in Section 5.

## 4. Analysis

We formulate the connected vehicle environment using set notation (see the following part).

*Summary of Symbols*

$V$: Set of vehicles, $V = \{v_1, v_2, \ldots, v_{|V|}\}$

$S$: Set of services, $S = \{s_1, s_2, \ldots, s_{|S|}\}$

$F$: Set of fog servers, $F = \{f_1, f_2, \ldots, f_{|F|}\}$

$R_{f_j}$: Set of vehicles which are located in coverage of a fog server $f_j$, $R_{f_j} \subset V$

$SQ_{f_j}$: Set of services that will be treated by a fog server $f_j$, $SQ_{f_j} = \{sq_{f_j,1}, sq_{f_j,2}, \ldots, sq_{f_j,|SQ_{f_j}|}\}$, $\forall sq_{f_j,k} \in S$

$L_{v_i}$: A timer of a vehicle $v_i$ for the current service, $L_{v_i}(t) = \alpha_{v_i} A_{v_i}$, $L_{v_i}(t+1) = L_{v_i}(t) - 1$

$E_{v_i}$: The expected delay of a current service for vehicle $v_i$, $E_{v_i} = H_{v_i} A_{v_i}$

Because of the dynamic nature of the mobile environment, elements of some sets change over time. For example, the elements of $R_{f_i}(t_m)$ are different from $R_{f_i}(t_n)$; that is, $R_{f_i}(t_m) \neq R_{f_i}(t_n)$, where $m \neq n$. Each vehicle requires services out of $S$, according to a Poisson process. The services provided by fog servers are typically location sensitive, for example, driving information such as local traffic congestion information or accident reports. On the other hand, cloud servers typically support entertainment services like multimedia streaming. Fog servers maintain a service queue to provide services to vehicles which are located in their local communication area.

When a fog server has unexpectedly failed, the controller recognizes the failure situation and performs the recovery process. The controller informs the neighboring fog servers, providing them with the service queued at the failed fog server at the time of failure. The neighboring fog servers then reschedule their service queues to recover from the failure. If vehicles are not located in the coverage area of any neighboring fog servers, boundary vehicles between the two neighboring fog servers serve as relay nodes.

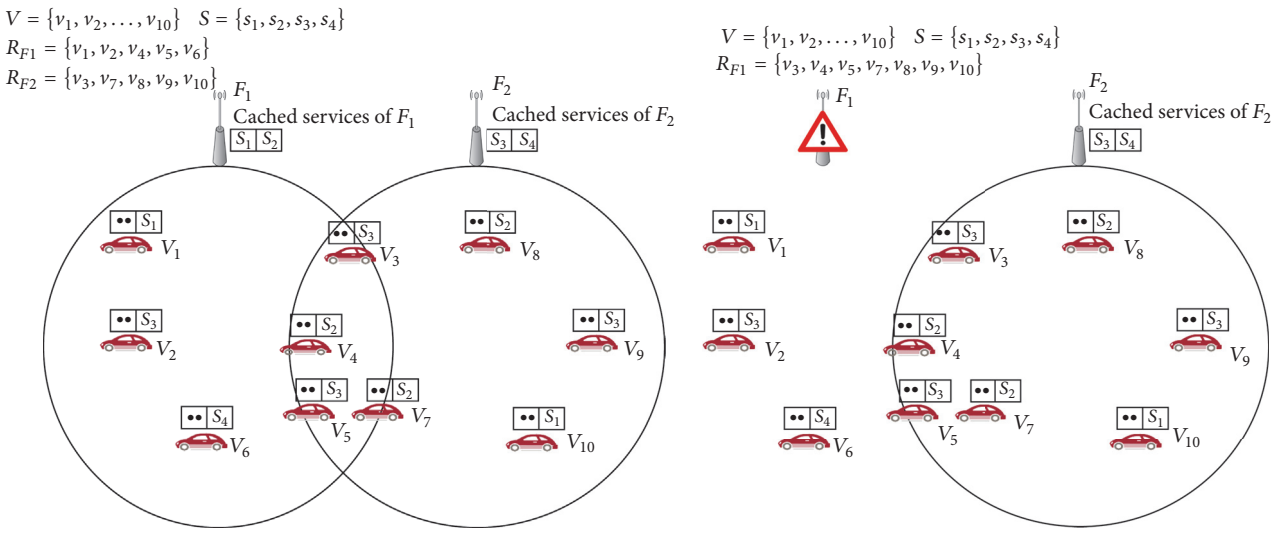
*4.1. Problem Description.* The objective function of the recovery process is shown in (1). Its purpose is to maximize the number of services provided to vehicles, while satisfying the delay requirement of each service, until the end of recovery time $t_e$.

$$\underset{V,S,F}{\text{minimize}} \quad \sum_{t=t_0}^{t_e} \sum_{i=0}^{|V|} I_{v_i}(t). \tag{1}$$

$I_{v_i}(t)$ is a binary variable which can have one value out of 1 and 0. It denotes whether the delay requirement of a service for vehicle $v_i$ is met or not at time $t$. A vehicle sets a timer $L_{v_i}$ at the time at which it requests a service. As in (2), $I_{v_i}(t)$ is changed to 1 when the timer expires. Therefore, the minimization of the total sum of $I_{v_i}(t)$ in (1) is equal to the maximization of the number of services vehicles can use.

$$I_{v_i}(t) = \begin{cases} 1, & \text{If } L_{v_i}(t) = -1, \\ 0, & \text{Otherwise.} \end{cases} \tag{2}$$

Equation (3) is a constraint to decide the candidate services to be recovered from the fog server failure. This means the service whose timer should be greater than the expected delay $E_{v_i}$ at time $t$. The notations $E_{v_i}$ and $L_{v_i}$ vary with a specific environment and the service type for each vehicle. The values of these two notations are decided by several determinants. First, $E_{v_i}$ is decided by two factors, $H_{v_i}$ and $A_{v_i}$, hop counts and delay of a service in one hop distance. $H_{v_i}$ is set to 1 if vehicles are in the coverage of the fog server. Otherwise, vehicles need a hop count greater than 1 to communicate with the available fog server via multihop. In addition, the hop count is influenced by the origin of the service, which can be a fog server or a cloud server. In other words, if a current service for a vehicle is not cached at the fog server, the service is retrieved from the cloud server, resulting in the increase of $H_{v_i}$. Because $A_{v_i}$ is a services delay in one hop distance, we multiply $A_{v_i}$ by $H_{v_i}$. Second, an initial value of the timer $L_{v_i}$ is a threshold of a services delay that users can allow. Thus, this timer is initialized by the multiplication of the one hop delay $A_{v_i}$ and the allowable hop count $\alpha_{v_i}$.




(a) Before the fog server $F_1$ failure

(b) After the fog server $F_1$ failure

Figure 2: An example of the recovery process.

For the performance comparison in Section 5, $\alpha_{v_i}$ is randomly selected.

$$\text{subject to} \quad L_{v_i}(t) - E_{v_i}(t) > 0, \quad E_{v_i}(t) > 0. \tag{3}$$

*4.2. Example of Recovery Process.* Figure 2 depicts an example of the recovery process. There are ten vehicles, four available services, and two fog servers. Each fog server is providing services to vehicles which are located in its communication coverage, as shown in Figure 2(a). Figure 2(b) depicts the situation when fog server $F_1$ fails. We identify two recovery cases. Any lost connections that can be recovered by connecting to a neighboring fog server are taken over by that fog server. In the example, when $F_1$ fails, vehicles $v_4$ and $v_5$ are served instead by fog server $F_2$. Any vehicles which cannot connect directly to another fog server must depend on neighboring vehicles to relay the connection to a neighboring fog server. In the example, vehicles $v_1$, $v_2$, and $v_6$ are needed to receive relayed data from boundary vehicles $V_3$, $V_4$, and $v_5$.

Each fog server makes a table to decide the priority of the services in the recovery situation. When the controller informs a certain fog server failure, neighboring fog servers start the recovery process by rescheduling queues for all the services including the broken fog servers services. Table 1 represents the service queues of fog server $F_2$ just after fog server $F_1$ fails. To make effective scheduling, the fog server maintains several attributes such as service ID, vehicle ID, service type, timer, and expected delay. First, the service types are divided into two cases according to which servers are responsible for offering a service to a vehicle. If a cloud server is responsible for providing a certain service, the service type is marked as $C$. On the other hand, the service type is marked as $F$ when a fog server offers a service. Second, each of these two cases, $C$ and $F$, is divided into two subcases, depending on whether the service is relayed by other vehicles or not. For instance, if a service is originated from a cloud server and it is relayed by not any infrastructure node but another vehicle, the service is marked as $C$ and $V$ at the same time. By utilizing these attributes that the fog server maintains, we apply a real-time scheduling scheme to find the optimal ordering of services.

Table 1: An example of the fog server's service queue.

| Vehicle ID | Service ID | Service type | Timer | Expected delay |
|---|---|---|---|---|
| $v_1$ | $s_1$ | $C, V$ | 9 | 7 |
| $v_2$ | $s_3$ | $C, V$ | 18 | 6 |
| $v_3$ | $s_3$ | $F$ | 10 | 2 |
| $v_4$ | $s_2$ | $C$ | 18 | 10 |
| $v_5$ | $s_3$ | $F$ | 16 | 2 |
| $v_6$ | $s_4$ | $F, V$ | 15 | 15 |
| $v_7$ | $s_2$ | $C$ | 12 | 10 |
| $v_8$ | $s_2$ | $C$ | 14 | 10 |
| $v_9$ | $s_3$ | $F$ | 18 | 2 |
| $v_{10}$ | $s_1$ | $C$ | 9 | 5 |

*4.3. The Proposed Real-Time Scheduling Algorithms.* Real-time scheduling has been studied for many fields of application, especially in MANET. Because the connected vehicle environment is similar to MANET in terms of dynamic features of its mobile environment, the proposed method is modified from the precedence real-time scheduling algorithms [11, 12]. One of such algorithms is Earliest Deadline First (EDF), which is a dynamic priority algorithm. When the fog server must decide the next order of service, it uses EDF to select the service for which the deadline is soonest. Another reasonable algorithm is Most Requested First (MRF). When applying the MRF, the fog server schedules services in the order of frequency of their request. Schedules are maintained before the start of the recovery process.

In order to make optimal scheduling, we propose a real-time scheduling algorithm that exploits precedence algorithms and attributes in the fog servers service queues. By doing this, we can obtain near-optimal results with the various requirements of delay. First, we adopt the EDF algorithm to serve as many services as possible until the recovery process is completed. We then add a decision method which selects a service according to its service type and one hop delay of a service, $A_{v_i}$. If the service type is marked as $F$ and the value of $A_{v_i}$ is small, the service can be provided to users in a short time. When the delay requirement of most services is tight, giving a high priority to a service which has the shorter service time is rational since it can maximize the total number of services meeting the deadline. Generally, the services provided by a fog server have shorter service time than the services from a cloud server; thus, the services from a fog server have higher priority in the recovery process. This policy makes sense in the respect that the service by the fog server is more critical than the service by the cloud server. The information for safe driving is provided from the fog server.

## 5. Simulation

We performed the simulation to evaluate the proposed idea in two cases. First, the control overhead and the connection recovery are evaluated in the connected vehicle environment. Then, the recovery from the fog server failure is evaluated as a numerical result in Table 3.

We constructed the campus network shown in Figure 3, using the Simulation of Urban Mobility (SUMO) [13] simulator. The mobile nodes drive along the black line according to their routing path. We generated nodes that have one mobility pattern out of the three patterns in Figure 1: random, predictable, and quasi-stationary. The beacon message period of each node is based on its mobility pattern; that is, nodes having random movement patterns send beacon messages more frequently than the others. We assume that the controller knows all the nodes mobility pattern, while Ku et al. [8] make all SDN wireless nodes send periodic beacon messages. Because our fog servers are controlled by an SDN controller, we utilized the Mininet-WiFi [14] simulator which considers both the SDN and the mobile environment.

*5.1. Control Message Overhead.* The result for the control overhead is shown in Figure 4. The control overhead consists of two types of messages: first the beacon messages that each mobile node periodically informs to an SDN controller and second the messages exchanged between mobile nodes whenever they meet each other, in order to collect neighboring information. This neighboring information is included in the NSI of each node, and this NSI is contained in the beacon message reported from each node to the SDN controller. These two types of control messages were considered in the simulation. The ratio of nodes having the random, predictable, and quasi-stationary mobility pattern is 1 : 1 : 1. Our method reduces the number of control messages by approximately 55% as compared with Ku et al.'s method [8], while still maintaining fresh topology. This is because the

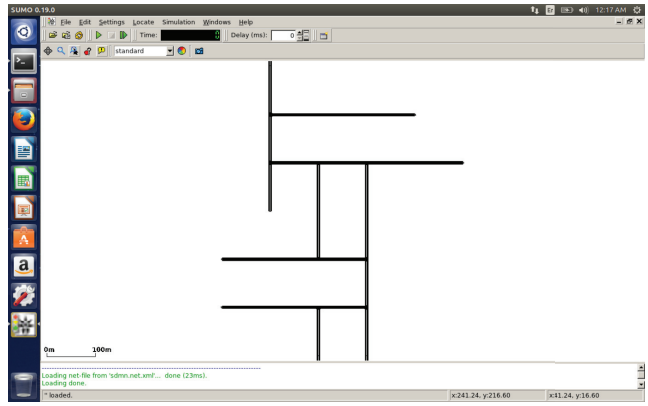


Figure 3: Building a campus network.

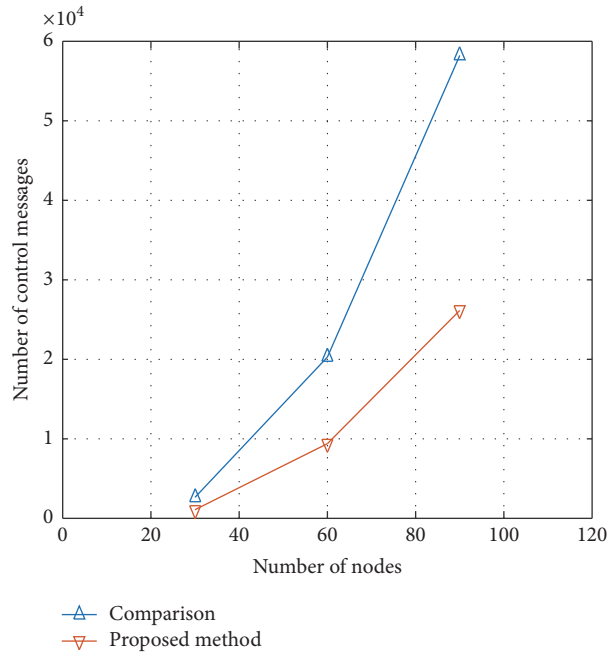



Figure 4: Control overhead.

controller can reduce each node control message period if it can predict its next location.

Figure 5 depicts the control overhead with different ratios of three mobility pattern nodes: random, predictable, and quasi-stationary. The ratios are set to 2 : 1 : 1, 1 : 2 : 1, and 1 : 1 : 2, respectively. The number of control messages is the greatest when the ratio is 1 : 2 : 1. This is because the nodes with predictable movement have a higher probability of encountering other nodes while driving along the regular routing path. In our simulation, the nodes with predictable movement are generally public buses. They move along the main road in the campus, so they can meet more vehicles than other types of nodes. On the other hand, because stationary nodes meet the least number of other nodes, the number of control messages is the lowest when the ratio is 1 : 1 : 2. Relative to the comparison, our method reduced control message

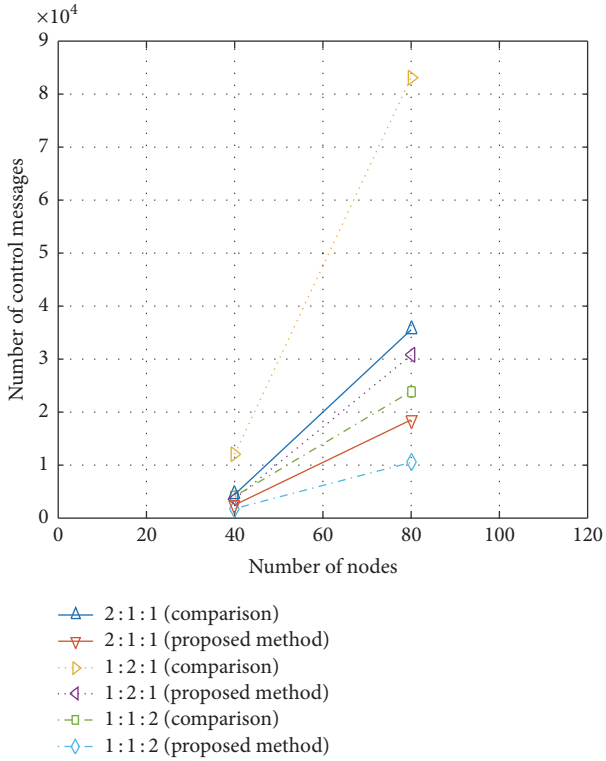



Figure 5: Control overhead in various environments.

Table 2: Evaluation parameters.

| Parameter | Value |
|---|---|
| Hop count from the fog server | 1 |
| Hop count from the cloud | 5 |
| Expected increasing hop count (random mobility) | 4 |
| Expected increasing hop count (predictable mobility) | 2 |
| Size of services of $S$ | $\{1, 2, 2, 3\}$ |

Table 3: The number of successfully delivered services.

| | EDF | MRF | Proposed algorithm | Optimal |
|---|---|---|---|---|
| Rigid delay | 5 | 7 | 7 | 7 |
| Sufficient delay | 9 | 8 | 9 | 9 |

overhead by 64%, 56%, and 47%, for mobility pattern ratios of 1 : 2 : 1, 1 : 1 : 2, and 2 : 1 : 1.

*5.2. Connection Lost Time.* The duration while the connection is lost is compared in Figure 6. As the number of nodes grows, the total failure time also increases. The result shows that the connection lost time is reduced by about 5% relative to the comparison method. Furthermore, the proposed method avoids wasting resources by just waiting for a while until the connection is automatically recovered, not starting a new routing process. It can predict when the connection will be automatically recovered using the information of the node moving pattern.

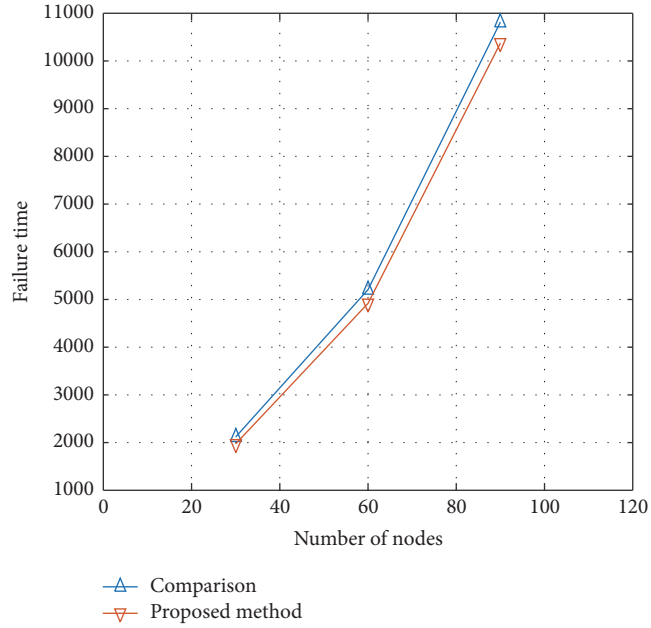



Figure 6: Connection lost time.

*5.3. Service Deadline Miss Ratio.* To evaluate our real-time scheduling algorithm, we set several parameters to the values shown in Table 2. After a fog server failure, the vehicles residing out of the fog server suffer from the long delay due to the multihop communication. To reflect this problem in the evaluation, we set the expected increasing hop count depending on the vehicles mobility pattern. The fog server which is responsible for recovery cannot find out the exact increased hop count, especially when the vehicles drive with random mobility. Therefore, the expected increasing hop count of vehicles which have random mobility is set higher than that of vehicles having predictable mobility. Depending on an initial value of the timer which is affected by $\alpha_{v_i}$, the evaluation result is differentiated. We classify two cases which have different requirements of delay by assigning the value of $\alpha_{v_i}$, rigid and sufficient delay cases. In the rigid delay case, $\alpha_{v_i}$ is randomly generated from integers over the range $[5 \cdots 9]$. Because users want to enjoy services in time, the timer is decided to the value which is close to the expected delay. This means that $\alpha_{v_i}$ is set close to $H_{v_i}$. The minimum value that users require is bounded to the expected hop count where the service type is $C$. The upper bound is decided to the worst case where the service type is $C$ and $V$ with the randomly moving vehicle. On the other hand, $\alpha_{v_i}$ is set to 10 in the sufficient delay case.

We compare our algorithm with EDF and MRF. Table 3 represents the total number of services which are provided successfully during the recovery. Since the number of services is 4 in the example, the possible scheduling cases are 24. Among these cases, there are some cases which have the maximum number of services. The optimal scheduling is one of them and their number of succeeded services is the optimal result.

The optimal result is 7 in the rigid delay environment. The proposed algorithm and the MRF show an optimal result, while EDF does not. This is because EDF only considers the deadline of the service. If many vehicles require tight delay of services, the strategy of offering many services at once is efficient like MRF.

In the sufficient delay environment, the EDF scheme outperforms the MRF, because the EDF is better than the MRF in maximizing the service throughput. Our algorithm shows the optimal result in both cases, because it considers services which have tight delay requirements while maximizing service throughput.

Based on the evaluation result, we observe two important comprehensions for rescheduling the service during the fog server recovery. First, we have to maximize service throughput while keeping delay requirements. Second, due to the services requiring low latency, it is reasonable to give a high priority to the service which will be quickly completed.

## 6. Conclusion

This paper proposed a method utilizing the network state information (i.e., the nodes mobility patterns and link quality) to overcome unstable communication in the fog computing and SDN-based connected vehicle environment. We built a campus network using the SUMO simulator and implemented the mobile SDN environment by modifying the Mininet-WiFi. Our simulation substantiated the fact that our method reduced the number of control messages by approximately 55% over the existing method. The proposed connection recovery scheme reduced failure time by about 5%. We further suggested a real-time scheduling algorithm to maximize the number of services successfully delivered to users even in a fog server failure situation.

As a future work, we will simulate a more complex situation of the fog server failure by using SUMO simulator, considering dynamic features in the connected vehicle environment such as congestion and mobility pattern in multihop communication.

## Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

## Acknowledgments

This work was supported by Mid-Career Researcher Program through NRF grant funded by the MEST (2016R1A2B4016588).

## References

[1] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. We, and L. Sun, "Fog computing: focusing on mobile users at the edge," https://arxiv.org/abs/1502.01815v3.

[2] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the ACM Workshop on Mobile Big Data (Mobidata '15)*, pp. 37–42, Hangzhou, China, June 2015.

[3] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[4] J. G. Andrews, S. Buzzi, W. Choi et al., "What will 5G be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[5] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: a viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[6] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: unbridling SDNs," in *Proceedings of the 1st European Workshop on Software Defined Networks (EWSDN '12)*, pp. 1–6, October 2012.

[7] T. Chen, M. Matinmikko, X. Chen, X. Zhou, and P. Ahokangas, "Software defined mobile networks: concept, survey, and research directions," *IEEE Communications Magazine*, vol. 53, no. 11, pp. 126–133, 2015.

[8] I. Ku, Y. Lu, and M. Gerla, "Software-defined mobile cloud: architecture, services and use cases," in *Proceedings of the 10th International Wireless Communications and Mobile Computing Conference (IWCMC '14)*, Nicosia, Cyprus, August 2014.

[9] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, "Software defined networking-based vehicular Adhoc Network with Fog Computing," in *Proceedings of the 14th IFIP/IEEE International Symposium on Integrated Network Management (IM '15)*, pp. 1202–1207, Ottawa, Canada, May 2015.

[10] OpenFlow Switch Specification, Version 1.1.0 Implemented (Wire Protocol 0x02), 2011.

[11] K. Liu, J. K. Y. Ng, V. C. S. Lee, S. H. Son, and I. Stojmenovic, "Cooperative data scheduling in hybrid vehicular ad hoc networks: VANET as a software defined network," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1759–1773, 2016.

[12] B. Dezfouli, M. Radi, and O. Chipara, "Mobility-aware real-time scheduling for low-power wireless networks," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM '16)*, San Francisco, Calif, USA, April 2016.

[13] Simulation of Urban Mobility (SUMO), http://sumo-sim.org/.

[14] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, "Mininet-WiFi: emulating software-defined wireless networks," in *Proceedings of the 11th International Conference on Network and Service Management (CNSM '15)*, pp. 384–389, Barcelona, Spain, November 2015.