Handover and Load Balancing for Distributed Network Control: Applications in ITS Message Dissemination

Anuj Kaul, Li Xue, Katia Obraczka, Mateus Augusto Silva Santos, Thierry Turletti

Handover and Load Balancing for Distributed Network Control: Applications in ITS Message Dissemination

Anuj Kaul, Li Xue, Katia Obraczka
UC Santa Cruz
{anujkaul,katia}@soe.ucsc.edu
xueli@ucsc.edu

Mateus A. S. Santos
Ericsson Research, Brazil
mateus.santos@ericsson.com

Thierry Turletti
Université Côte d'Azur, Inria
thierry.turletti@inria.fr

***Abstract*—In this paper, we build upon our prior work on D2-ITS, a flexible and extensible framework to dynamically distribute network control to enable message dissemination in Intelligent Transport Systems (ITS), and extend it with handover and load balancing capabilities. More specifically, D2-ITS' new handover feature allows a controller to automatically "delegate" control of a vehicle to another controller as the vehicle moves. Control delegation can also be used as a way to balance load among controllers and ensure that required application quality of service is maintained. We showcase D2-ITS' handover and load-balancing features using the Mininet-Wifi network simulator/emulator. Our preliminary experiments show D2-ITS' ability to seamlessly handover control of vehicles as they move.**

***Index Terms*—intelligent transport systems, software defined networking, distributed network control, next-generation vehicular network, handover**

I. Introduction

According to the European Union's 2010/40/EU Directive [1], Intelligent Transport Systems (ITS) are defined as "systems in which information and communication technologies are applied in the field of road transport, including infrastructure, vehicles and users, and in traffic management and mobility management, as well as for interfaces with other modes of transport". As such, ITS is pushing the envelope and establishing next frontiers in a number of information and communication technology, including vehicular networks (VANETs) and vehicular communication. In the context of ITS applications, vehicles can exchange safety information, e.g., to assist drivers in avoiding accidents, to perform autonomous- or self-driving tasks, to coordinate traffic flow, communicate road conditions, etc. Furthermore, information flow can also support "infotainment" services, navigation, etc. [2]. For instance, the European Telecommunications Standards Institute (ETSI) defined multiple messages to support ITS services such as Road Hazard Warning [3].

In VANETs, there are two main types of communications, namely Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I). As vehicles get equipped with increasingly more computation- and storage capabilities as well as more complex sensors (e.g., cameras, radars, etc.), vehicular communication will require not only more resources from the underlying communication infrastructure, but also also more stringent quality-of-service guarantees (e.g., latency, bandwidth, etc). In order to support growing vehicular communication demands, new technologies such as Dedicated Short Range Communications (DSRC) [4], which enables V2V communication, have been proposed to ensure that stringent requirements (e.g., extremely low latency messages for autonomous driving) can be met. As such, VANETs will likely employ heterogeneous underlying technologies including multiple radio access technologies, such as IEEE 802.11, IEEE802.11p, and cellular technologies such as Long Term Evolution (LTE) and its enhancements towards fifth-generation (5G) networks.

As ITS applications become more diverse and demand more resources and as new communication technologies become available, VANET's network infrastructure is faced with considerable challenges, i.e., accommodate underlying heterogeneous communication technologies, while satisfying ever more demanding applications. Software Defined Networking (SDN) [5], through control and data plane separation and centralized intelligence, is considered a key technology enabler to achieve flexible, programmable, and high-performing next-generation vehicular networks [6], [7]. Its application to VANETs has been proposed in recent years. However, logically centralized control function or static control distribution are not compatible with VANET's scalability, geographic distribution, administrative decentralization, and diverse quality-of-service requirements. In [8], we propose the Dynamic Distributed Network Control for ITS, or D2-ITS, a flexible and dynamically distributed network control framework, which aims at addressing the scalability, heterogeneity, delay intolerance, and decentralization requirements of ITS applications. To this end, D2-ITS' distributed control plane: (1) is decoupled from the data plane, and (2) is based on a control hierarchy that can dynamically adjust to current environment and network conditions in order to support ITS application requirements.

In this paper, we build upon our D2-ITS framework and extend it with handover and load balancing capabilities. More specifically, D2-ITS' new handover feature allows a controller to automatically "delegate" control of a vehicle to another controller as the vehicle moves. Control delegation can also be used as a way to balance load among controllers and ensure that required application quality of service is maintained. We showcase D2-ITS' handover and load-balancing features using

the Mininet-Wifi network simulator/emulator [9].

The remainder of the paper is organized as follows. Section II reviews the related works while Section III provides a brief description of the D2-ITS architecture and model. In Section IV, we discuss different types of handover and their trade offs, and in Section V present D2-ITS's handover and load balancing features, including their algorithms and implementation. Section VI describes our experimental methodology, experiments we conducted and their results. Finally, Section VII concludes and discusses directions for future work.

II. Related Work

Controlling and managing network resources in VANETs is a challenging and complex task. Abundant literature is available on theoretical research (e.g., [6], [7], [10]).

There has also been several efforts that conduct experimental evaluation of new technologies to support VANETs. For example, the work in [11] presents VANET emulation experiments in Mininet-WiFi [9] to show the applicability and benefits of SDN to VANETs. They define a new implementation setup for the vehicle to enable more realistic experiments. The proposed setup consists of combining two Mininet-WiFi "objects", namely station and switch, which emulate end-devices and switches, respectively. It enables the use of heterogeneous wireless interfaces which can support V2V and V2I communication coupled with a software switch that can support programmable flow management by a centralized SDN controller. DeVANET [12] proposes a decentralized SDN-based VANET architecture where each domain is controlled by a controller. However, DeVANET does not consider dynamic control distribution in response to application needs and current network and environment conditions. Another SDN-based centralized control approach for VANETs is presented in [13]. Experiments based on the OpenNet simulator using POX as the SDN controller compare performance against a traditional network control plane (i.e., not using SDN) in terms of packet delivery ratio, throughout, and packet delay.

Though not specifically applied to VANETs, the work presented in [14] describes a load balancing handover scheme performed by a centralized controller which selects APs clients should associate with as they move. The proposed handover mechanism is demonstrated using Mininet-Wifi experiments.

As previously pointed out, in [8], we propose D2-ITS, a novel dynamically distributed network control plane framework and show how it is able to support ITS' Road Hazard Warning (RHW) message dissemination application.

III. D2-ITS

As illustrated in Figure 1, D2-ITS' control plane is decoupled from the data plane (following the SDN model), however it can be distributed across multiple controllers according to a control hierarchy. In this particular example, controllers at the highest level of the D2-ITS hierarchy are located at cellular base stations (RBS), whereas second-level controllers are located at road-side units (RSUs). As shown in the figure, RSUs are controlling vehicles in Domains A and B, whereas Domain C vehicles are being controlled by the controller in one of the RBSs. These "local" controllers which are located closer to devices being controlled and their physical environment are able to react much faster to underlying dynamics besides maintaining topology knowledge through periodic updates from controlled devices.

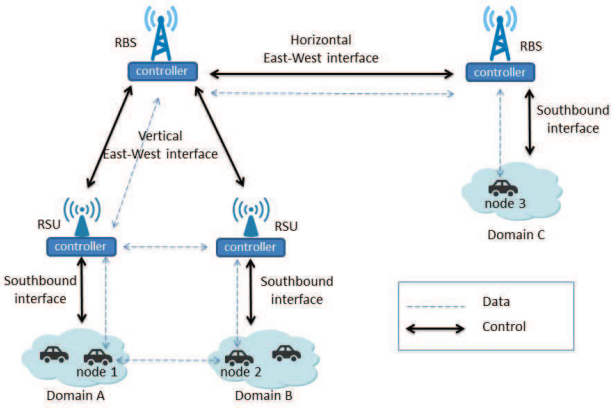

Fig. 1. D2-ITS Architecture.

A. Control Plane

Controller-to-controller communication happens through the *east-west interface* and is essential for control state synchronization, topology discovery and maintenance, controller- and vehicle authentication, etc. In the case of ITS' Road Hazard Warning (RHW) service, an event-based road message dissemination service specified by the European Telecommunications Standards Institute (ETSI), Cooperative Awareness Messages (CAMs), which may carry measurement reports, vehicle information (e.g., location, speed), and vehicle authentication information, are also part of the control plane [1]. Figure 2 illustrates how CAM messages are processed by controllers.

The control plane also includes controller-to-switch communication which, in SDN terminology, is referred to as the *southbound interface*. Through the southbound interface, controllers send data forwarding rules to the data forwarding devices (i.e., vehicles in our ITS scenario) they control.

B. Data Plane

Decoupled from the control pane, D2-ITS' data plane focuses solely on forwarding data messages using "rules" installed in forwarding devices by the controller. In general, ITS applications will involve Vehicle-to-Vehicle (V2V) communication, Vehicle-to-Infrastructure (V2I) / Infrastructure-to-Vehicle (I2V) communication, Infrastructure-to-Infrastructure (I2I) communication, or a combination thereof.

- Vehicle-to-Vehicle Data Plane: Data can be forwarded between vehicles through V2V direct connection or through

[1]More information on ETSI Intelligent Transport Systems standards is available at http://www.etsi.org/technologies-clusters/technologies/intelligent-transport.

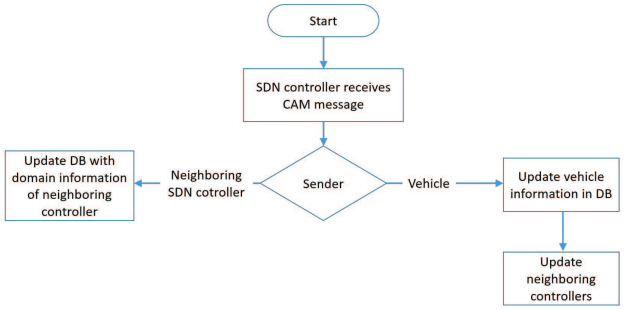

Fig. 2. D2-ITS controller's processing of CAM messages

devices which are part of the data forwarding infrastructure, e.g., Road-Side Units (RSUs) using V2I, I2I and I2V communication. For example, in the case of autonomous driving communication, messages from vehicles can be directly delivered to other vehicles in the domain. On the other hand, as mentioned in [8] and as illustrated in Figure 1, if Node 1 in Domain A needs to communicate with Node 2 in Domain B, Domain A's RSU can relay traffic to Domain B's RSU (I2I communication), which will then deliver it to Node 2.

- Vehicle-to-Infrastructure (V2I) / Infrastructure-to-Vehicle (I2V) Data Plane: In ITS, the infrastructure itself may originate messages, e.g., to communicate road conditions, alerts, etc. ITS' Road Hazard Warning (RHW) service defines a variety of events including traffic jams, hazardous conditions and messages that will be generated to alert road users (e.g., cars, trucks) of such events For example, Decentralized Environmental Notification Messages (DENMs) [3] are mainly used to provide the necessary alerts in the case of emergency situations (e.g., eminent risk of collision) or warnings (e.g., road congestion conditions). As such, DENMs should be conveyed through the communication infrastructure and delivered to road users in the geographic area affected by the event, also called "relevance area" [3].

Figure 3 illustrates the processing of DENM messages by the controller.

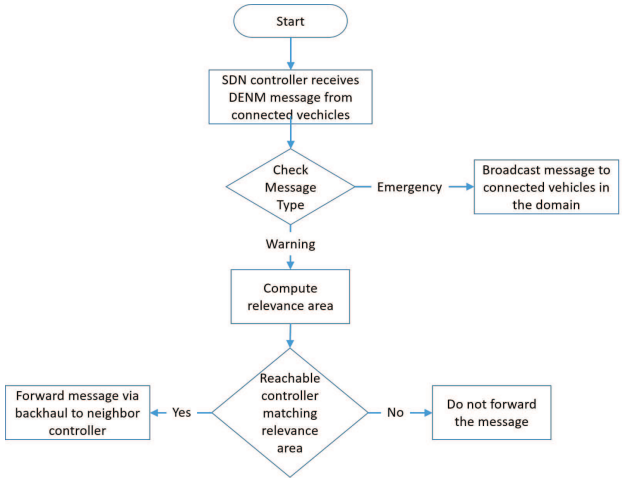

Fig. 3. D2-ITS controller's processing of DENM messages

IV. D2-ITS Handover and Load Balancing

In [8], we demonstrated that by decoupling the control-from the data plane, D2-ITS leverages network programmability to address ITS scalability, delay intolerance and decentralization. In this work, we extend D2-ITS with handover and load balancing capabilities in order maintain continuous service availability and satisfy application quality-of-service requirements when subject to conditions such as unpredictable mobility patterns of fast moving vehicles, communication channel impairments, etc.

A. Handover

Communication networks that support user- or device mobility should be able to provide continuous service availability while users/devices are moving. In infrastructure-based networks, e.g. cellular networks or wireless LANs (WLANs), a handover (or handoff) refers to the process of transferring an ongoing call or data session from the current infrastructure device to which the mobile is associated (e.g., cellular base station, a WLAN access point, or AP) to another infrastructure device. Handovers can occur when the mobile device moves away from its current network point of attachment and thus out of its transmission range and gets closer to another. Handovers can also occur for load balancing, e.g., when an AP is near- or at capacity and decides to offload current or incoming associations to a nearby, less loaded AP. This usually improves call quality and user experience. In VANETs, vehicles may move at relatively high speeds (e.g., compared to pedestrians) and also have limited degrees of freedom due to the fact that they need to use roadways [15]. In D2-ITS, handovers happen when a controller delegates control of a vehicle to another controller due to vehicle mobility or in order to balance controller load.

In general, handovers can be classified according to different criteria. For example, depending on when they are initiated, handovers are either "hard" or "soft":

- **Hard Handover:** In this case, the association with the current controller is released either before or as the association with the new controller is established. As such, disruption in connectivity may occur which may result in data loss.
- **Soft Handover:** In soft handover, before breaking the previous connection to the old controller, a new connection to the new controller is established and active sessions are transfered to the new controller. Once the active sessions are transfered, the connection to the old controller is broken.

Handovers can also be classified depending on who initiates them, as follows:

- In **controller-initiated handovers**, the controller decides to initiate the handover of some of its connected vehicles

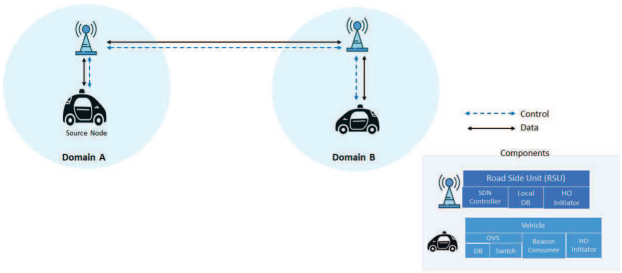

Fig. 4. RHW communication using gateways

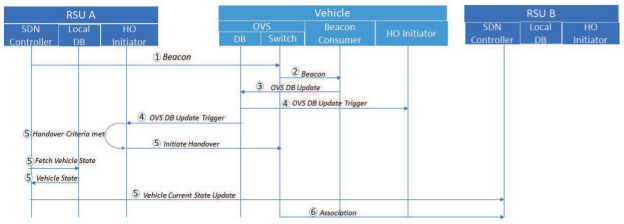

Fig. 5. Handover initiated by controller

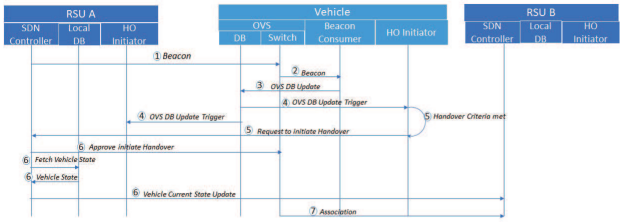

Fig. 6. Vehicle-initiated handover

to another controller, either at the same hierarchical level (vertical delegation), or across hierarchical levels (horizontal delegation). For example, depending on its current load, a controller can decide to handover one or more vehicles that it currently controls (e.g., it can choose vehicles currently generating high traffic load) to a neighboring controller.

- In **vehicle-initiated handovers**, the vehicle itself decides to initiate the handover to a neighboring controller. For example, depending upon the signal strength or transmission errors, a vehicle may decide to associate itself to a neighboring controller with bettwr signal quality. However, for security purposes, controllers need to authenticate vehicles in order to authorize the handover.

B. Handover Steps

The handover process can be divided into 3 phases:

- **Controller discovery:** In D2-ITS, controllers broadcast *beacon* messages periodically. *Beacons* contain information about the controller, including IP address, port number, current load, etc. Once a vehicle receives a *beacon* from a controller, it calculates the corresponding Received Signal Strength Indicato (RSSI) value. Vehicles store this information in their local database and also propagate it to the controller to which it is connected. The controller or the vehicle can use RRSI information to decide whether a handover should be initiated.
- **Controller selection:** Using the information in *beacons*, different criteria (e.g., RSSI value as discussed above) can be used to select the handover's target controller. Once the controller is selected, either the controller or the vehicle can initiate the handover.
- **Association:** In D2-ITS, once the target controller for the handover is selected, the vehicle proceeds with the association. Also, the current controller can send the current state it has about the vehicle to the next controller. This helps reduce or eliminate the authentication process of the vehicle by the target controller.

V. D2-ITS Handover Design and Implementation

Figure 4 shows a simple handover scenario with two RSU controllers, each of which controlling a vehicle. The figure also illustrates D2-ITS' software components present in the controller and vehicle. As shown in Figure 5, the D2-ITS controller builds upon an SDN controller and uses a local database to store state information (information about currently controlled nodes, neighboring controllers, etc), as well as a *handover initiator*, which is described in more detail in Section V-C below. The vehicle itself runs a software switch, a local database, and, in order to handle handovers, also includes a *beacon consumer* and *handover intiator*, which are also described in Section V-C.

A. Controller-Initiated Handover

Figure 5 illustrates the workflow of controller-initiated handover as well as D2-ITS' software components present in the controller and vehicle. In this particular scenario, RSU A is controlling a vehicle which is about to be handed over to RSU B. The sequence of steps illustrated in Figure 5 is as follows:

1 RSU A's controller periodically broadcasts *beacon* messages.
2 Upon receiving *beacon* messages from the controller, the vehicle's software switch forwards them to the vehicle's *beacon consumer* application.
3 The *beacon consumer* processes the *beacon* and updates this information in the vehicle's local database.
4 These updates generate an update notification to the vehicle's and the controller's (in this case the controller in RSU A) *handover initiator*.
5 When the handover criteria of RSU A's *handover initiator* are met, the controller selects the target controller

for the handover and initiates the handover request to the vehicle. The controller also fetches the vehicle's state information from its local database and sends it to the new controller, in this caseo, RSU B.

6 Upon receiving the handover request from its controller, the switch inside the vehicle starts to initiate link establishment (or association) with the new controller (RSU B).

B. Vehicle-Initiated Handover

Figure 6 illustrates the vehicle-initiated handover workflow.

1-4 : Similar to controller-initiated handover as explained above.

5 When the handover criteria of Handover application within vehicle are met, switch within the vehicle sends the request to initiate handover to the controller.

6 After controller A validates the handover request, it sends approve handover request.Controller also fetches the vehicle information from its local database and sends vehicle state information to the new controller of RSU B. This is again similar to the controller initiated handover scenario.

7 After the switch inside the vehicle receives approve handover request from controller, switch starts to initiate link establishment.

C. Handover Extensions to D2-ITS

To achieve seamless handover in D2-ITS, we included the following extensions to D2-ITS's original design (as described in [8]):

- **Beacons:** For the purpose of controller discovery by vehicles, controllers periodically send *beacon* messages which are heard by vehicles in the proximity of the controller. Note that, while some protocols like IEEE 802.11 already generate periodic beacon messages, others such as LTE do not.
- **Vehicle database:** To store information related to neighboring controllers, we use a database in the vehicle where we keep the following information about known controllers:
  - RSSI value;
  - Controller IP address;
  - Controller port number;
  - Time when last beacon was received.
- **Beacon-consumer application:** Beacon consumer application would co-locate with OVS inside the vehicle. The functionality of the beacon consumer application is to consume the beacon messages from neighboring controllers and to update the table described above.
- **Handover-initiator application:** There would be two different handover-initiator applications:
  - Handover-initiator application at vehicle: This application would reside in the vehicle and would listen to the updates on the ovsdb table. Specific sets of criteria to initiate the handover are configured in this application. When those criteria are met, it will initiate the handover.
  - Handover-initiator application at controller: This application would reside in the controller and would listen to the updates on the ovsdb table remotely. Similar to the Handover-initiator application at vehicle, specific configurable handover criteria are set in this application and, when they are met, handover is triggered.

VI. Experimental Evaluation

The goal of this simulation is to investigate and compare handover latency and packet loss of D2-ITS and centralized ITS architecture. In order to do so, we need set up vehicle-infrastructure-vehicle end-to-end communication, then we could time stamp at the moment sending the message and receiving the message to obtain message transmission latency due to handover and also count the packet loss for performance. From one step begins, we implement following three basic functions: inter-controller negotiation, vehicle to controller communication and handover while vehicles moving. More work are needed in the future.

A. D2-ITS Handover Prototype

As the basis for our implementation, we use the *Pox* [16] open-source controller and the OVS software switch [17].

We used Mininet-WiFi[2] as our experimental platform. More specifically, we ran our experiments with Mininet-WiFi version 2.2.1d1 running on Ubuntu version 16.4 in VirtualBox 5.2.6. Mininet-WiFi [18], [9] is a network emulator which can emulate different types of wireless nodes, such as access points, eNodeBs, and wireless links. It also supports node mobility and provides visualization capabilities.

B. Experiments and Results

The main goal of our experiments is to demonstrate D2-ITS' support for handover and load balancing in a seamless fashion. In particular, we use ITS Road Hazard Warning (RHW) message dissemination application to show that D2-ITS does not incur significant additional latency nor causes service disruptions.

Our experiments demonstrated the following D2-ITS functionality: The implementation takes advantage of the POX framework, leveraging some of its components and adding the missing functionalities. The functions used for the East-West Interface are categorized as follows.

- Messenger Module: A POX module providing a message queue (MessageNexus). Messages are sent to the channels and are formatted in JSON. It is possible to define channels, send and receive messages, and be notified when a connection is established. The messenger.tcp_transport module allows the message queue to open a TCP socket to listen for messages coming from outside the controller. A thread binds a socket to a port and listens for new connections. A new thread is spawned for each connection to

[2]https://github.com/intrig-unicamp/mininet-wifi

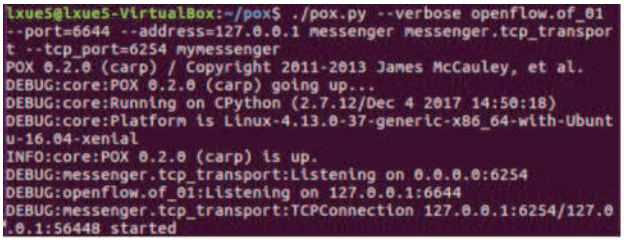

Fig. 7. Starting the Messenger Module

send and receive messages and interact with the message queue. Threads are handled via the POX rococo library (based on Python threading).

- East_West_Messenger Module: A Python module implementing the communication for the East-West Interface and the vehicles. It uses event handlers to trigger actions when a new controller or the user client running on the cars connect, when a message is received or when a message needs to be sent. The communication between controllers, East-West Interface, uses a handshake kind protocol where the controllers can exchange point to point or broadcast messages. The communication between a vehicle and its current controller, uses a channel to send messages, CarLink, and a channel to poll for messages for the vehicle, CarLinkRelay. The communication is initiated by the vehicle that this way can be used for vehicles to send messages, more details are described in the following subsection.
- Link_Client Module: An independent script used to interact with the controllers. It uses a TCP socket to open a connection to the messenger running in POX (IP address and port are known) and then provides a CLI to send messages and display the ones received. Messages are in JSON, they include the channel name, and the actual message.

The messenger module and east_west_messenger module are defined in mymessenger.py, which needs to be put in the ext/ directory of POX controller in order to initialize the connection. Moreover, the controller to controller communication happens with a controller invoking the test_client program to send a message to another controller at any time. The communication between controllers uses a specific channel.The messenger function can be initiated through following command:

- *mininet-wifi/$ cd pox*
- *mininet-wifi/pox/$ ./pox.py --verbose openflow.of_01 --port=6644 --address=127.0.0.1 messenger messenger.tcp_transport --tcp_port=6254 mymessenger*

Figure 8 shows how to start the client module, running on another terminal in the virtual machine. In the future the model of a vehicle will trigger this client controller. For example, a vehicle sending an emergence message will trigger the controller to communicate with the neighbor controller. Here is the command to start the POX client controller with the test_client.py.

Fig. 8. Starting the Client Module

- *mininet-wifi/$cd pox*
- *mininet-wifi/pox$./pox/messenger/test_client.py 127.0.0.1 6254*

C. Vehicle to Controller Communication

In vehicle to controller communication, as first step, a vehicle need to automatically discovery and authenticate with RSU during its moving without a need for manual configurations. For example, while the vehicle roaming to a new location, the vehicle could reactively or proactively discovery the RSU in the domain through active scan or bgscan. As mentioned before, controller is co-located with RSU, thus controller will be selected in case of RSU is selected. When a vehicle have selected and successfully associate with the current domain RSU, RSU send the co-located controller's information(i.g.,IP address, MAC address) embedded in the beacon frames to the vehicle for deciding the controller handover. We assume if the vehicle is in the range of more than one RSUs, it can select the RSU and its controller by strongest signal, defined in RSSI parameter. We experienced significant issues in getting Mininet-WiFi to activate the POX controller through script. At the end we decided to work-around the issues through the following extensions: we extended the POX controller to receive messages coming from vehicles through a socket interface. An external program (car_client) is designed for stateless communication between vehicles and controllers, which is invocable from Mininet-WiFi. Car_client takes some arguments, including a message to send, the vehicle name, and the IP address and port number of the controller to which to send the message. In order to send a message to the associated controller, the program opens a socket with the specified controller, sends the message and closes the socket(stateless communication). The program can also poll the controller to check if there are messages from the controller to the vehicle waiting to be delivered. Additionally, there are following functions in east-west_messenger module defined to support vehicle to controller communication. The communication between a vehicle and its current controller, uses a channel(CarLink) to send messages, and a different channel(CarLinkReplay) to poll for messages for the vehicle. The communication is designed to be initiated by the vehicle that this way can be more realistic. The following functions to are invoked:

- Initialization: It defines and starts the functions to enable communication between the controller and vehicles.
- Car Link Relay(CarLinkRelay, CarLinkRelayBot): This function keeps track of the connections sending messages

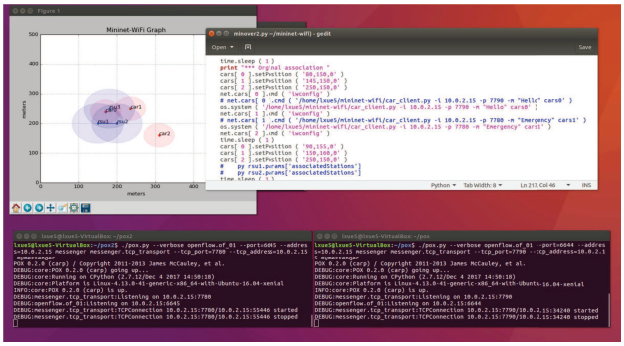

Fig. 9. Vehicle to Controller Communication

from vehicles. For each connection on the channel, it keeps a message counter, listens to messages and triggers actions and replies(ACKs). In addition, this function sends messages when triggered by controller events.

- Car_Client Module: An independent python script used by a vehicle to interacts with the controller. It is stateless and uses a TCP socket to open a connection to the messenger running in POX (IP address and port are known). It provides its ID, the car name, can send a message and/or request messages queued for it. Messages are in JSON, include the ID, car name, the channel name, carlink or calink_request, and the eventual message, such as emergence.

The steps to simulate vehicle to controller communication are shown in Figure 9 as follows: We run multiple remote POX controllers, each in its own Xterm. Due to all the POX controllers running inside the same virtual machine, each controller is identified by a different port number. Inside the Mininet-WiFi car definition, we invoke the car_client program whenever a message has to be sent to a controller, specifying the controller associated with the RSU the vehicle is currently associated with. Periodically the vehicle invoking the car_client program queries the controller co-located with the RSU that the vehicle is currently associated with to check if there are messages waiting to be delivered. We believe that a stateless communication approach is more appropriate than keeping up persistent TCP connections in a mobility environment where associations of car to RSU / controller is dynamic and continuously changing.

D. Handover

We consider two scenarios to observe the handover:

- There are two RSUs (RSU1 and RSU2) deployed which have very small signal overlap area, while 3 cars deployed, shown in Figure 10.
- There are three RSUs (RSU1 and RSU2 and RSU3) deployed which have large signal overlap area, while 3 cars deployed, shown in Figure 11.

The handover implementation has 4 stages: scanning, RSU selection, RSU association, Controller association, alternatively, data packet detour. At this stage, we simulate the

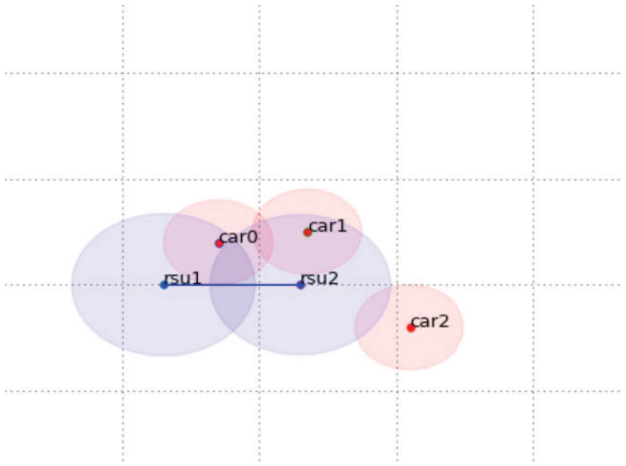

Fig. 10. Handover Scenario 1

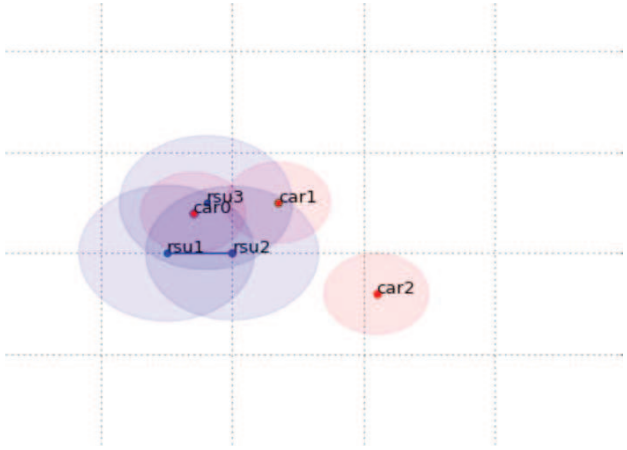

Fig. 11. Handover Scenario 2

handover through Mininet-WiFi ssf (Strong Signal First) association based on RSSI value. We configure the environment as follows: enables the wmediumd, uses UserAP module for RSU rather than OVSKernel AP, enable bgscan mechanism. In order to enable the controller association while the moving vehicle achieving association with RSU, we set the controller as a parameter for RSU. Thus, when a vehicle association with specific RSU, the vehicle also will obtain the information of the controller. This information should be invoked inside the wireless packet. In this experiment, we directly configure the controller address in the vehicle node to mimic this behavior. After association, the vehicle can communication with the controller based on mechanism descried aforementioned. Figure 12 shows the behavior of the activity of association while vehicles reset the new position.

VII. Conclusion and Future Work

In this paper, we built upon our prior work on D2-ITS, a flexible and extensible distributed network control framework that addresses stringent requirements of emerging and future applications, such as Intelligent Transportation Services (ITS),

Fig. 12. Handover Observation

and extend it with handover and load balancing capabilities. D2-ITS' new handover feature allows a controller to automatically "delegate" control of a vehicle to another controller under certain conditios, e.g., as the vehicle moves. Control delegation can also be used as a way to balance load among controllers and ensure that required application quality of service is maintained. We showcase D2-ITS' handover and load-balancing features using the Mininet-Wifi network simulator/emulator [9]. As part of our future work, we plan to demonstrate D2-ITS' handover and load balancing features in a wide-range of realistic scenarios and considering other handover criteria such as controller load, application latency and bandwidth requirements, etc.

ACKNOWLEDGMENTS

This work is part of the DrIVE Associated Team between Inria, Unicamp, Ericsson Research and UCSC. It has been partially funded by Inria's Associated Team Program, the French ANR Investments for the Future Program under grant ANR-11-LABX-0031-01, and the US National Science Foundation under project CNS 1321151.

REFERENCES

[1] Directive 2010/40/eu of the european parliament and of the council of 7 july 2010 eur-lex.europa.eu. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:207:0001:0013:EN:PDF

[2] T. Taleb, E. Sakhaee, A. Jamalipour, K. Hashimoto, N. Kato, and Y. Nemoto, "A stable routing protocol to support its services in vanet networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 6, pp. 3337–3347, Nov 2007.

[3] ETSI, "Intelligent transport systems; vehicular communications; basic set of applications; specifications of decentralized environmental notification basic service," ETSI, Tech. Rep. EN 302 637-3 V1.2.1, 2014.

[4] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, July 2011.

[5] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, 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, Jan 2015.

[6] K. Zheng, L. Hou, H. Meng, Q. Zheng, N. Lu, and L. Lei, "Soft-defined heterogeneous vehicular network: architecture and challenges," *IEEE Network*, vol. 30, no. 4, pp. 72–80, July 2016.

[7] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Software-defined networking for rsu clouds in support of the internet of vehicles," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 133–144, April 2015.

[8] A. Kaul, K. Obraczka, M. A. S. Santos, C. E. Rothenberg, and T. Turletti, "Dynamically distributed network control for message dissemination in its," in *2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, Oct 2017, pp. 1–9.

[9] R. Fontes and C. Rothenberg. The user manual, mininet-wifi emulator for software-defined wireless networks. [Online]. Available: https://github.com/intrig-unicamp/mininet-wifi

[10] D. K. Sheet, O. Kaiwartya, A. H. Abdullah, and A. N. Hassan, "Location information verification cum security using tbm in geocast routing," vol. 70, pp. 219–225, 12 2015.

[11] R. D. R. Fontes, C. Campolo, C. E. Rothenberg, and A. Molinaro, "From theory to experimental evaluation: Resource management in software-defined vehicular networks," *IEEE Access*, vol. 5, pp. 3069–3076, 2017.

[12] A. Kazmi, M. A. Khan, and M. U. Akram, "Devanet: Decentralized software-defined vanet architecture," in *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, April 2016, pp. 42–47.

[13] S. Thun and C. Saivichit, "Performance improvement of vehicular ad hoc network environment by cooperation between sdn/openflow controller and ieee 802.11p," *Journal of Telecommunication Electronic and Computer Engineering*, vol. 9, no. 2-6, 2017.

[14] N. Kiran, Y. Changchuan, and Z. Akram, "Ap load balance based handover in software defined wifi systems," in *2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, Sept 2016, pp. 6–11.

[15] J. Harri, F. Filali, and C. Bonnet, "Mobility models for vehicular ad hoc networks: a survey and taxonomy," *IEEE Communications Surveys Tutorials*, vol. 11, no. 4, pp. 19–41, Fourth 2009.

[16] Pox controller. [Online]. Available: https://github.com/noxrepo/pox

[17] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of open vSwitch," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'15. Berkeley, CA, USA: USENIX Association, 2015, pp. 117–130.

[18] 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 *2015 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 384–389.