Vehicle Software Updates Distribution with SDN and Cloud Computing

Meysam Azizian, Soumaya Cherkaoui, and Abdelhakim Senhaji Hafid

The authors propose an architecture for distributing software updates on vehicles based on software defined networking and cloud computing. They show that using SDN, the emergent networking paradigm, which provides on-demand network programmability, adds substantial flexibly for deploying software updates on vehicles.

ABSTRACT

Vehicles have embedded software dedicated to diverse functionality ranging from driving assistance to entertainment. Vehicle manufacturers often need to perform updates on software installed on vehicles. Software updates can either be pushed by the manufacturer to install fixes, or be requested by vehicle owners to upgrade some functionality. We propose an architecture for distributing software updates on vehicles based on SDN and cloud computing. We show that using SDN, the emergent networking paradigm, which provides on-demand network programmability, adds substantial flexibility for deploying software updates on vehicles. We propose solutions for how vehicular networks can be modeled as connectivity graphs that can be used as input for the SDN architecture. After constructing graphs, we present an SDN-based solution where different frequency bands are assigned to different graph edges to improve the network performance.

INTRODUCTION

Vehicular technology has evolved tremendously during the last 15 years, making vehicles safer, more intelligent, and more pleasant to drive. These advancements have been made possible by embedding intelligence within onboard systems through extensive software coded features. With the role played by software ever expanding in modern vehicles, a new challenge has emerged. Manufacturers are facing the necessity to upload software updates into vehicles either to fix bugs or to improve existing functionality. To do so, customers are usually required to go for service at their dealers. A single trip for service can be inconvenient for customers, but as software components become preponderant in vehicles and the need for updates more frequent, this can become impractical. Recently, several manufacturers have shown interest in new ways of uploading software updates over the air through either Wi-Fi, cellular, or satellite connections. They envision that vehicles will eventually become serviceable remotely just like smartphones are now, with fixes, updates, and new features added over the air.

Ford, which previously delivered updates to its infotainment system with physical USB uploads, has geared up to using Wi-Fi Internet connections for its recent models, and plans to use satellite connections in the future. To download an update, vehicles need to be in range of a Wi-Fi access point at home or elsewhere for the duration of the software upload. Often, this may be either infeasible or impractical. For another manufacturer, Tesla, vehicles receive software update notifications to add new features and functionality over cellular connections. However, the manufacturer advises owners to use Wi-Fi for faster downloads.

With vehicular dedicated short-range communications (DSRC) [1] expected to be available in vehicles very soon, a new opportunity arises to dynamically update software on vehicles by using vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications. Apart from DRSC, LTE-Direct, which has been in development in recent years in fourth and fifth generation (4G/5G) fora, is also a promising technology for V2V communications in future. In this article, we show how software defined networking (SDN), paired with vehicular communications and cloud computing, can be used to add substantial flexibility for deploying software updates on vehicles. This flexible programmability can be very useful for deploying non-critical software updates (e.g., infotainment) on the fly, without the need to go for service.

SDN has emerged as a powerful networking paradigm that can provide scalable and flexible means to manage networks. With SDN, new protocols can be deployed easily, and a broad spectrum of embedded networking functions modified and manipulated. SDN decouples the data plane and the control plane so that forwarding functions and network functions are dissociated. An SDN-based architecture is composed of two main components: the SDN controller and SDN devices. The SDN controller, as the logically centralized intelligence of the SDN network, can control programmable SDN devices' behavior through a standardized southbound interface protocol called OpenFlow. SDN makes it possible to have unified management of different types of SDN devices regardless of the SDN device vendor [2]. This powerful paradigm works in a simple way. Each SDN device has a flow table with several fields specifying how an incoming packet should be treated. The SDN controller modifies flow tables either proactively, depending on application layer needs and network topology, or reactively, based on traffic-dependent on-demand requests from SDN devices. SDN devices transfer packets based on polices dictated to each of them from the SDN

Digital Object Identifier:
10.1109/MCOM.2017.1601161

Meysam Azizian and Soumaya Cherkaoui are with the Univeristy of Sherbrooke; Abdelhakim Senhaji Hafid is with the University of Montreal.

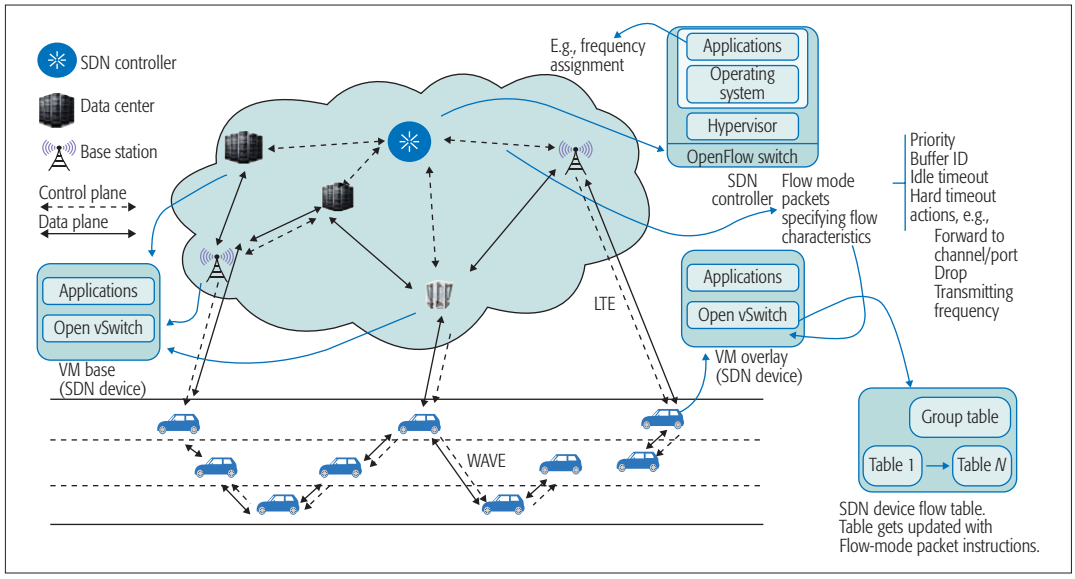

Figure 1. SVC system architecture. The SDN controller instructs all SDN devices (switches) on how flows should be treated.

controller, which has global knowledge about SDN devices; hence, there is no need for hardware configuration.

In order to deploy software updates on vehicles, we propose an SDN-based vehicular cloud architecture (SVC) that leverages V2V communications. In SVC, vehicles are considered SDN devices, and SDN controllers can be located on roadside units (RSUs) or base stations (BSs), reside at the edge of the network, or be located at servers in a data center. Software update using SVC can be considered as a way to leverage SDN for deploying the future Internet of Things (IoT) [3]. V2V communication allows vehicles to cooperate in software update distribution in a similar way to device-to-device (D2D) cooperation. Using SDN with vehicular communications poses two important challenges, though. First, different from wired networks, vehicular networks are prone to frequent topology changes as vehicles move close to/away from each other. A swift and autonomous neighbor discovery mechanism needs to be devised to allow the SDN controller to acquire and maintain a global view of network topology. To address this, we propose a solution to construct vehicles' connectivity graphs based on standard vehicular beaconing conveying position information among vehicles. Vehicles communicate with each other in a distributed way and provide connectivity information to the SDN controller, which constructs the connectivity graphs. Second, V2V communication is known to suffer considerably from interference and hidden node problems. In order to mitigate these problems, we propose that the SDN controller, as a centralized manager of the network, execute a novel scheme that uses mathematical optimization to assign different operating frequencies to vehicles (SDN devices). The frequency assignment makes it possible for vehicles to communicate in the data plane in a multihop manner while fully resolving the hidden node problem. Furthermore, the scheme allows co-channel and adjacent channel interferences to be sizably reduced.

System Architecture and Assumptions

In SVC architecture, we suppose that vehicles, which are SDN devices, are equipped with OpenFlow-capable software switches such as Open vSwitch. We also suppose that SDN devices are capable of virtualization and can host virtual machines (VMs). A VM will be able to install software updates on the vehicle. Here, a software update is considered as a service. When a vehicle is interested in the software update service, the vehicle VM will be replicated at the nearest datacenter which offers that service [4, 5]. RSUs, BSs, and other edge equipment can host micro-data centers and participate in fog computing [4] to deliver services, including software updates to vehicles. They can also be SDN devices or host SDN controllers. The vehicle VM and data center VMs are named VM overlay and VM base, respectively [6]. A VM overlay migrates to other nearby VM bases when it is close to moving out of the communication range of the current VM base. The SDN controller is responsible for managing these VM replications and migrations. SDN controllers are located in some of the data centers or edge equipment, and are connected with RSUs, BSs, and other data centers, which form a cloud together [7]. Therefore, the control plane decisions are not entirely made by a centralized element; several cloud elements can collaborate to make decisions. Figure 1 illustrates SVC architecture.

In SVC, vehicles are equipped with V2V communication technology, and some vehicles may have LTE connections. Vehicles that have LTE connections or are in the communication range of infrastructure elements (e.g., RSUs) transfer the information about their connectivity with other vehicles to the SDN controller. The SDN controller uses this information to construct the vehicles' connectivity graph model in order to update flow tables and allocate frequency bands. DSRC designates four 10 MHz frequency bands, called service channels (SCHs), to be used for service data communications. These frequency bands will be allocated by the SDN controller for SDN data plane software update transfers. One of these fre-

SDN-controllers are located in some of the data centers or edge equipment, and are connected with RSUs, BSs and other data centers which form a cloud together. Therefore the control plane decisions are not entirely taken by a centralized element; several cloud elements can collaborate to take decisions.

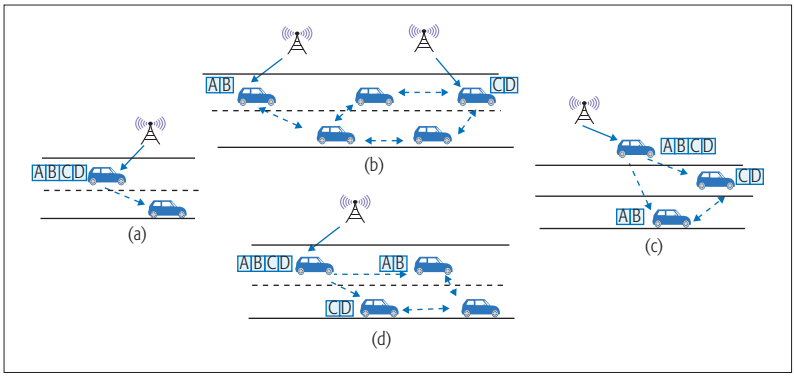

Figure 2. Content distribution: a) the left-most vehicle receives the content $[A|B|C|D]$ and shares it with the neighbor vehicle; b) the left-most and right-most vehicles receive half of the content and share it with their neighboring vehicles. In c) and d), the left side vehicles receive the content and split it before delivery to neighbors. In the aforementioned cases, vehicles collaborate with each other to get the complete content based on the provided instructions from the SDN controller. Therefore, software update distribution with SDN improves network performance by decreasing both used cellular bandwidth and the corresponding usage fee (and also DSRC bandwidth), and software update delivery delay.

quencies is also chosen as the de facto frequency for SDN control plane signalling.

As input to the connectivity graph calculation and frequency assignment, the SDN controller receives information from vehicles regarding their location, received power, and relative mobility with their neighbors. This information can be obtained by each vehicle using standard safety message beacons exchanged in the DSRC control channel (CCH) [1]. Vehicles use the SDN control plane to convey this information to the SDN controller either directly or in a multihop manner.

SDN devices have flow tables, and the SDN controller updates these tables in the control plane via flow-mode packets in order to route software updates to interested vehicles. Service (software updates) advertisement happens in a CCH period as specified by the WAVE standard. Therefore, whenever an update is available and a vehicle is interested, it switches to the SDN control plane signaling channel to send information and receive flow table updates from the SDN controller.

The SDN controller updates the flow tables regularly following a change in topology to avoid the connection loss while a vehicle is receiving a service. However, when the SDN controller cannot update switches due to unexpected situations (e.g., sudden speed change of vehicles or communication problems due to channel fading), vehicles temporarily use other routing protocols, such as general packet switched radio (GPSR), as a backup to avoid connection loss [8]. The occurrence of flow table updates is decided by the SDN controller based on the stability of the constructed graphs. The SDN controller estimates the stability of the constructed graphs by measuring relative mobility between vehicles, and updates the flow tables based on that estimation.

Connectivity Graph Model

To construct the connectivity graph model, SDN controllers need vehicle positions and their calculated relative mobility. To calculate relative mobility, we use a similar method to the one proposed in [9]. Each vehicle calculates the delays of two consecutive standard beacons received from a neighbor to determine the relative mobility with that neighbor.

The use of beacon delays as a metric to calculate the relative mobility is superior to other metrics such as speed or position. Indeed, other metrics may not clearly represent channel conditions such as fading or communication obstacles. For example, two neighbor vehicles in nominal communication range may have almost the same speed, but the channel condition between the two might be highly faded; this can make it impossible for them to communicate.

Each vehicle encapsulates its coordinates and transmission time in standard beacon messages. Therefore, the receiving vehicle can know the transmitter location and calculate the delay of transmission. A vehicle can calculate the relative mobility of a neighbor when it receives the second beacon message from that neighbor. Therefore, each vehicle can construct relative mobility tables and send them to the SDN controller via the control plane, directly or in a multihop manner; this information can be appended to beacon messages to avoid extra messages. Relative mobility is calculated at each node using a logarithm form function represented as: 10 multiplied by log base 10 of (x/y), where the nominator, x, is the new calculated beacon delay, and the denominator, y, is the old calculated one.

After receiving relative mobility information, the SDN controller constructs the directed connectivity graph model. In the graph, each edge represents a connection from one vehicle to another, while vertices represent vehicles. For each edge, the SDN controller examines the corresponding relative mobility value. The SDN controller compares the relative mobility value with a threshold to see if it is strong enough to be considered as a communication link. When communication links have been determined, the SDN controller calculates the routes and updates the corresponding flow tables of SDN devices. Based on the relative mobility corresponding to each edge in the graph, the SDN controller also decides on the frequency for updating flow tables and assigning channels.

For route calculation, and content distribution, the SDN controller can follow several strategies. When a software update is not in high demand, such as a paid new feature, the SDN controller can route the update directly to the interested vehicle using the shortest path. When the software update is popular, the SDN controller can orchestrate delivery by determining clusters of interested vehicles. In each cluster, each vehicle receives part of the software update, and they cooperate to assemble it [10]. This is illustrated in Fig. 2. Another interesting technique the SDN controller can use is distributed caching [11]. Indeed, when a software update is available, it is likely to be requested by a large number of vehicles of the same make. Distributed storage of these updates enhances the opportunities for vehicles' collaboration to access the software update solely through V2V communication.

It is worth noting that the proposed connectivity graph model calculation can be enhanced by using other data. In addition to standard information available from beacons, several other parameters could potentially be used, when available, to construct the connectivity graph and improve its

stability. Examples of these parameters include the estimated vehicle trajectory, which can be acquired from the vehicle navigation system, real-time traffic conditions of roads, and travel history of vehicles. However, these data might not always be obtainable. Furthermore, using such extra parameters can come at the expense of extra overhead.

Frequency Assignment

V2V communication is prone to interference and hidden node problems, which cause frequent collisions in the communication channel. Collisions at the medium access control (MAC) layer cause higher delays and lower throughputs. In order to mitigate these problems, we propose that the SDN controller assign to vehicles (SDN devices), different operating frequencies to be used in the data plane.

Once the connectivity graph model is constructed, it can be used by the SDN controller to assign different frequency bands to be used by vehicles for communicating over the different edges of the constructed graph. The SDN controller, with its global view on the network, plays a centralized controller role in assigning different frequencies to vehicles in order to degrade interference levels and solve hidden node problems.

The SDN controller assigns frequencies in such a way that no two neighbor edges in the nominal communication range of each other use the same frequency band. Therefore, co-channel interference will be highly degraded for dense vehicle situations, or even eliminated in spare network conditions. Also, the SDN controller takes care of adjacent-channel interference by carefully choosing frequencies and assigning them to non-adjacent edges. Hence, adjacent channel interference will similarly be reduced, even if some of this interference is inevitable in highly dense scenarios.

The proposed frequency assignment scheme used by the SDN controller is based on a mathematical optimization model using binary integer programming (BIP). The mathematical optimization model calculates the maximum number of edges (i.e., transmission links) that can transmit (be activated) simultaneously in a single frequency band. The frequency assignment scheme uses the optimization model to recursively assign different frequency bands to different edges.

Optimization Model

As input to the optimization problem, the SDN controller uses the knowledge acquired through constructing the connectivity graph model. The input dataset comprises:

1. The power matrix representing the power received by each node from its neighbors
2. The edge matrix of the connectivity graph
3. The path matrix of the chosen paths toward the vehicle

In order to assign frequencies for different edges, the power matrix representing the received power by each node has to be examined. When vehicles are operating in the same frequency, communication over each edge is potentially interference to communications in other edges. The power matrix is therefore used to calculate the signal-to-interference-plus-noise ratio (SINR) on each edge. An edge in the connectivity graph can be active when the SINR of that edge is adequate; that is, it must pass a minimum threshold value to be considered for activation.

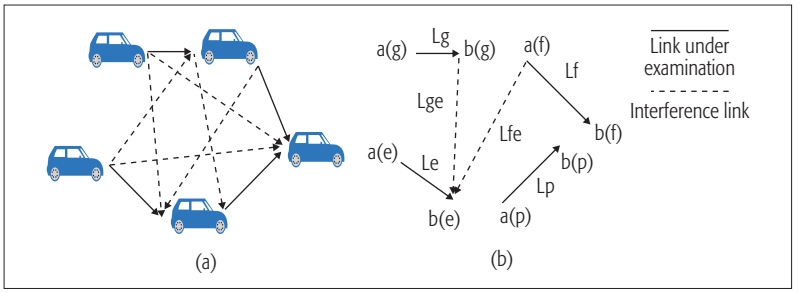

Figure 3. Co-channel interference modeling: a) edges in bold are links under examination by the SDN controller, the others are interference; b) illustration of the effect of interference on a link under examination by the SDN controller.

We write the optimization model as a BIP optimization where all variables are binary. The objective of the model is to find the maximum number of edges that can be active simultaneously in the same frequency. The edges in the objective are the ones that cause the minimum co-channel radio interference to each other. Therefore, the objective function to maximize can be represented as the sum of all edges, where edges are binary variables holding values zero or one, as in [12, Eq. 3]. An edge is assigned value one if it is active, and value zero otherwise. Maximizing the objective function is subject to three constraints.

The first constraint ensures that a vehicle can either transmit or receive at the same time. Therefore, only one edge can be activated at each node; this constraint is represented in in [12, Eq. 4]. The second constraint ensures that an edge can only be activated when the corresponding vehicle is transmitting; we assign value one to the binary variable of the vehicle when it is transmitting and zero otherwise; this constraint is represented by [12, Eq. 5]. The last constraint verifies that the SINR of an edge is strong enough to be considered for activation. *Signal* here stands for the signal strength of an edge under examination. *Interference* is signals from other interfering edges. *Noise* is noise power density. This constraint is represented in [12, Eq. 8]. Figure 3a shows an example of how co-channel interference affects communication links. Figure 3b illustrates the effect of interferences on a link of under examination (Le). We use the branch and bound method to solve this optimization model.

Assignment Scheme

In the DSRC standard, six channels are reserved for services [1]. However, only four of these channels (channels 174, 176, 180, and 182) are dedicated to general-purpose service use. The two remaining channels (channels 172 and 184) are reserved for "public safety applications involving safety of life and property" [13].

The proposed frequency assignment scheme uses the previously presented optimization model in successive rounds, up to four, based on the maximum number of frequency bands available in DSRC. The number of rounds can be less than four if vehicle density is low, and little interference exists. Each round comprises several steps.

First round:

Step 1: The SDN controller executes the optimization model and determines a set of edges that can transmit concurrently without interfering with each other.

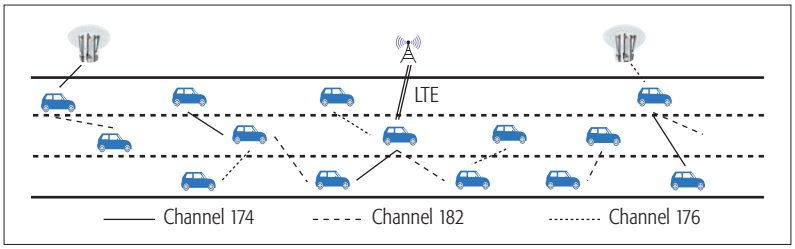

Figure 4. Frequency assignment example. The links assigned to channels 174,182, and 176 are calculated though the first, second, and third rounds of the proposed frequency assignment scheme, respectively. There is no need to proceed to the fourth round as the first three channels can cover the graphs' edge. The graphs are modeled through the proposed graph connectivity scheme.

Step 2: The SDN controller assigns the first channel, channel 174 (5.865 to 5.875 GHz), to this set of edges.

Step 3: The determined set of edges are removed from the input of the optimization model for the next round. This is done by setting the powers of those edges to zero in the new input power matrix.

After the frequency assignment of the first round, it is probable that some edges will be left without frequency assignment. This can be explained by two reasons:

1. Edges sharing the same vertex cannot be activated in the same round (i.e., assigned the same frequency).
2. An edge may have too low an SINR to be activated; some links need to be assigned a different frequency in order for the SINR to become high enough and the link activated in the second stage.

Second round: For the second time, the SDN controller executes the optimization model with the modified data set and gets a new set of edges (first step). This time, the SDN controller assigns channel 182 (5.905 to 5.915 GHz) to the generated set (second step). The reason for choosing this channel is to address adjacent channel interference: no neighbor frequency with the already assigned frequencies. Similar to the first round, the SDN controller updates the input dataset for the next round (third step).

Third round: If some edges are left without frequency assignment in the second round, they are considered in the third round. The round follows the same steps as in the second round with the difference that the assigned frequency band is that of channel 176 (5.875 to 5.885 GHz).

Fourth round: In this round, all edges that remain without frequency assignment in the third round are assigned to operate in channel 180 (5.895 to 5.905 GHz) without following any extra steps. The scheme stops proceeding here, as this is the last frequency band left. Vehicles operating in channel 180 simply use carrier sensing and random backoff delays to avoid collisions as specified in the WAVE standard. Figure 4 shows a vehicular network example where communication links have been assigned different frequency bands based on the proposed scheme.

With the proposed frequency assignment scheme, collisions and interference are reduced in the wireless medium. Therefore, the delay for vehicles to receive their services (software update) is shortened. Figure 5 shows a comparison in terms of the average delivery delay when using multiple frequencies assigned in SVC, and a scheme where vehicles use WAVE with the same DSRC frequency. In the simulations, all vehicles receive the software updates from their nearest data center.

With SVC, the average delay is smaller than with WAVE. This can be explained by several factors, including:

1. Vehicles using WAVE cause more mismanaged interference with each other.
2. Route discovery takes more time with WAVE alone than with SVC, and route recalculation is more frequent with WAVE alone.
3. The hidden node is more frequent with WAVE alone, causing collisions and retransmission delays.

Figure 5 shows that the delay gap between WAVE and SVC increases as the network gets denser. This is expected as the aforementioned problems get worse when density increases. Having a centralized SDN controller improves network performance by carefully managing interference, solving hidden node problems, and assigning adequate flow tables to SDN devices.

Open Research

Some fundamental and interesting research issues are still open to be able to fully realize the promise of the proposed SDN architecture. We have identified some of these areas that need particular attention from both academia and industry.

Incentives for Vehicles: Some incentives should be given to motivate vehicles that use their cellular networks to transfer neighbors' information to SDN controllers. Examples of these incentives include discounts on service, higher bandwidth [14], or free cellular connectivity. Similar kinds of incentives can be given to vehicles to encourage them to cooperate in content delivery (caching, etc.).

Quality of Service Satisfaction: Following frequency assignment, the SDN controller can optimize transmission scheduling in SVC based on several criteria including fairness or service differentiation. Both issues of fairness and service differentiation should be fully investigated given the scarcity of vehicular networking resources and the variety of software installed on vehicles. Fairness can relate to all vehicles having their fair amount of usage of network resources to receive software updates. The SDN controller can also consider different priorities for software updates, translating into different quality of service (QoS) needed for delivery. For example, a safety-related update can be prioritized over a software update that enhances infotainment system features.

Security Considerations: As the SDN controller in SVC has a global view over the SDN network, the network can be secured by re-instructing the SDN device flow tables at any time to mitigate an anomaly (e.g., instructions to drop some packets). The anomaly can be identified by the SDN controller security applications, based on the collected information from the SDN devices. However, security threats in SVC should be fully analyzed to design the corresponding security applications or even to restructure the proposed architecture. Several applications have already been proposed to deal with the security threats

of SDN architectures in general [15]; these applications can be adapted to be used in SVC. Some example of threats can be:

1. Attacks on the SDN controller may lead to the attacker controlling SDN devices' behavior in the data plane by manipulating flow rules.
2. Misbehavior of a cloud element: Since the SDN controller collaborates with cloud elements to define the control plane rules and make decisions, the cloud elements involved should be trusted.
3. Misbehavior of SDN devices: SDN devices that relay flow rules can be intelligently subversive and manipulate flow instructions.

There should be procedures for SDN devices to check the credibility of the received flow instructions.

Conclusion

In this article, we advocate for an architecture using SDN, cloud computing, and vehicular communications to distribute software updates to vehicles, and we show how such an architecture takes shape in SVC. SVC leverages vehicular communications and salient SDN features to distribute software updates in a flexible way to vehicles that need fixes or new features. SVC efficiently uses V2V beaconing information to construct the graphs needed by SDN controllers to control the network in a systematic way. SVC further makes effective use of networking resources by orchestrating channel use among participating vehicles. Some open issues make particularly interesting research topics to investigate in the future, including incentivizing vehicles to cooperate and bolstering the architecture with security features.

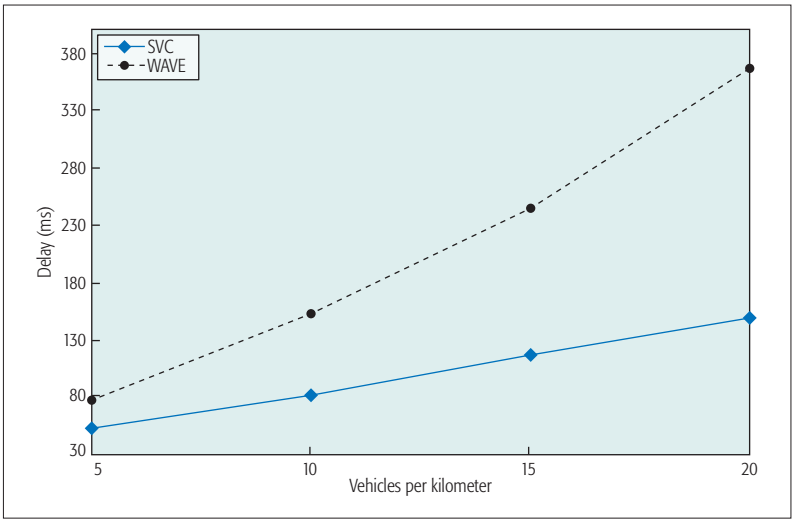

Figure 5. Comparison between SVC and using WAVE alone. The simulation scenario uses a four-lane highway with different vehicle densities. The maximum velocity of vehicles is 25 km/h. The data rate and transmission power for all vehicles are fixed to 6 Mb/s and 20 dBm, respectively. When using WAVE alone, the routing method used is ad hoc on-demand distance vector.

References

[1] R. A. Uzcátegui and G. Acosta-Marum, "WAVE: A Tutorial," *IEEE Commun. Mag.*, vol. 47, no. 5, May 2009, pp. 126–33.

[2] Z. He, J. Cao, and X. Liu, "SDVN: Enabling Rapid Network Innovation for Heterogeneous Vehicular Communication," *IEEE Network*, vol. 30, no. 4, July 2016, pp. 10–15.

[3] A. Al-Fuqaha *et al.*, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications", *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 3, 2015, pp. 2347–76.

[4] M. A. Salahuddin *et al.*, "RSU Cloud and Its Resource Management in Support of Enhanced Vehicular Applications," *IEEE GLOBECOM Wksp. Cloud Computing Systems, Networks, and Applications* , 2014, pp. 127–32.

[5] N. B.Truong, G. Lee, and Y. Ghamri-Doudane, "Software Defined Networking-Based Vehicular Adhoc Network with Fog Computing," *IFIP/IEEE Int'l. Symp. Integrated Network Management*, 2015, pp. 1202–07.

[6] R.Yu *et al.*, "Toward Cloud-based Vehicular Networks with Efficient Resource Management," *IEEE Network*, vol. 27, no. 5, Sept. 2013, pp. 48–55.

[7] Y. Xu and S. Mao, "A Survey of Mobile Cloud Computing for Rich Media Applications," *IEEE Wireless Commun.*, vol. 20, no. 3, June 2013, pp. 46–53.

[8] I. Ku *et al.*, "Towards Software-Defined VANET: Architecture and Services," *IEEE Int'l. Wksp. Mediterranean Ad Hoc Networking*, 2014, pp. 103–10.

[9] Z. Zhang, A. Boukereche, and R. Pazzi, "A Novel Multi Hop Clustering Scheme for Vehicular Ad-Hoc Networks," *ACM Int'l. Symp. Mobility Management and Wireless Access*, 2011, pp. 19–26.

[10] Y. Cao, J. Guo and Y. Wu, "SDN Enabled Content Distribution in Vehicular Networks," *13th IEEE Int'l. Conf. Innovative Computing Technology*, 2014, pp. 164–69.

[11] N. Golrezaei, P. Mansourifard, and A. F. Molisch, "Base-Station Assisted Device-to-Device Communications for High-Throughput Wireless Video Networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, 2014, pp. 3665–76.

[12] M. Azizian, S. Cherkaoui, and A. Senhaji Hafid, "A Distributed Cluster Based Transmission Scheduling in VANET," *IEEE ICC*, 2016.

[13] "FCC Code of Federation Regulations 47, Part 95," Personal Radio Services, 2009.

[14] H. Li, M. Dong, and K. Ota, "Control Plane Optimization in Software-Defined Vehicular Ad Hoc Networks," *IEEE Trans. Vehic. Tech.*, vol. 65, no. 10, 2016, pp. 7895–7904.

[15] I. Ahmad, S. Namal, and M. Ylianttila, "Security in Software Defined Networks: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 4, 2015, pp. 2317–46.

Biographies

Meysam Azizian received his M.Sc. degree in radio communication from Blekinge University, Sweden, in 2013. In 2012, he was a degree project student with the Department of Electric and Information Technology, Lund University. He is currently pursuing a Ph.D. degree in electrical and computer engineering with the INTERLAB, Université de Sherbrooke, Quebec, Canada. His current research interests are vehicular cloud computing, SDN, and resource sharing.

Soumaya Cherkaoui [M'99, SM'15] is a professor at the Department of Electrical and Computer Engineering, Université de Sherbrooke. Since 2000, she has been the director of Interlab, a research laboratory that conducts research funded by both government and industry. Before joining the Université de Sherbrooke as a faculty member, she worked for industry as a project leader on projects for the aerospace industry. She has co-authored over 200 publications in reputable journals and conferences in the area of communication networks. She has served as a General Chair, Technical Committee Chair, and Editor of many conferences and journals. She is a Professional Engineer in Quebec, Canada.

Abdelhakim Senhaji Hafid is a full professor at the University of Montreal, where he founded the Network Research Lab (NRL) in 2005. He is also a research fellow at CIRRELT. Prior to joining the University of Montreal, he spent several years as a senior research scientist at Telcordia Technologies (formerly Bell Communications Research), New Jersey, working on major research projects on the management of next generation networks including wireless and optical networks. He has extensive academic and industrial research experience in the management of next generation networks, QoS management, and communication protocols.