A Software Defined Network Architecture for GeoBroadcast in VANETs

Yu-Chun Liu\textsuperscript{1}, Chien Chen\textsuperscript{1,2} and Suchandra Chakraborty\textsuperscript{1}

\textsuperscript{1}Department of Computer Science and \textsuperscript{2}Information Technology Service Center
National Chiao Tung University
Hsinchu, Taiwan
friendshipsarea.cs00g@nctu.edu.tw; chienchen@cs.nctu.edu.tw; suchandratutul@gmail.com

Abstract—This paper proposes a Software Defined Network (SDN) architecture for GeoBroadcast in VANETs. We have implemented a component to automatically manage the geographical location of Road Side Units (RSUs), which are used as a basis for our GeoBroadcast routing. GeoBroadcast in a vehicular network supports periodic broadcast messages from a source vehicle to the destination vehicles that are located in a specific geographical region. In existing Intelligent Transport Systems (ITS), the GeoBroadcast mechanism can be implemented using traditional IP networking. Typically, every periodic warning messages received at nearest RSU from the source must be routed to the control center in ITS, where it is redirected to every other RSUs, that are located in the destination geographical region for broadcasting. As a result, huge overhead in the control center is produced and higher network bandwidth is consumed. However, in our SDN based GeoBroadcast mechanism, the first warning message received by the source RSU is sent to the SDN controller as a packet-in message. The SDN controller will decode the packet-in message and use topological and geographical information to set up the routing paths to the destination RSUs, by installing appropriate flow entries on the corresponding RSUs and intermediate switches, for the following periodic warning messages that are to be broadcasted. As per our simulation with OpenNet a significant reduction by 84\% in controller overhead, 60\% in network bandwidth consumption, and 81\% in latency are achieved.

Keywords—VANETs;SDN;RSU;GeoBroadcast; ITS

I. INTRODUCTION

Applications for Intelligent Transport Systems (ITS) aim to provide road users with improved traffic safety, traffic efficiency, and additional values in vehicular communication systems [1]. European Telecommunications Standards Institute (ETSI) has published GeoNetworking [2], which is a network-layer protocol for mobile ad hoc communication by utilizing geographical positions for distribution of information such as hazardous road warning messages. Geographic-based Broadcast (GeoBroadcast) is one of the message forwarding schemes of GeoNetworking in which a message is forwarded hop-by-hop (i.e. vehicle-by-vehicle) until it reaches the vehicles in the destination area determined by the message. Then, these vehicles will rebroadcast the message to cover the other vehicles within the destination area. Since Inter-Vehicle Communications (IVCs) may be broken frequently due to the variation in density and speed of vehicles on the roads. ETSI standard [3] proposes a combination of ITS center to Road Side Unit (RSU), vehicle to RSU (V2R/R2V) and vehicle to vehicle (V2V) communications for efficient delivery of messages to the destination area. Furthermore, ETSI [4] suggests conversion between GeoNetworking and IPv6, as the architecture of the ITS center is IP based, the GeoNetworking message is converted to an IP packet before sending it to ITS center. In [5-6], the conversion between GeoNetworking and IPv6 is used to implement a GeoBroadcasting (GBC) scheme called C2CNET. In C2CNET, the ITS center maintains RSUs’ location information and link status. Typically, RSU receives warning messages periodically and redirects all messages to the ITS center, where every warning message is processed and then forwarded to every other RSUs within the destination area. Individual processing and forwarding causes a huge overhead in the ITS center and higher network bandwidth consumption.

Therefore, we propose a Software Defined Network (SDN) [7] architecture for GeoBroadcasting in VANETs. SDN is based on a new abstraction of network architecture that decouples the control and data planes. In the control plane, a centralized controller controls network devices from multiple vendors. A standard protocol (e.g. OpenFlow) between control and data planes provides the programmability for operators to customize their applications. The OpenFlow protocol functions as follows. When an OpenFlow switch receives a packet for which no matching flow entry is installed in the flow table of the switch to instruct how to process the received packet, a “table-miss” occurs and the table-miss flow entry can be programmed so that the received packet is encapsulated in a packet-in message and forwarded to the controller. Packet-in message is used by the OpenFlow switch to send a replica of the ingress packet to the SDN controller, when there is no flowtable match to process this received packet. Then the controller makes a decision to instruct the switches how to process the rest of the packets in the same flow by sending flow modify messages to add flow entries on the switches. Each flow entry contains a match field and an action field. The packets that match, the match field of an existing flow entry will be processed according to its action field.
To take full advantage of SDN, we adopt Floodlight [9] OpenFlow controller to act as ITS control center. In this framework, the controller only needs to process the first broadcast message sent from the RSU nearest to the source to compute the routes for the rest of the messages. Our SDN based GeoBroadcast architecture is shown in Fig. 1, which includes the OpenFlow controller, OpenFlow switches and OpenFlow RSUs. OpenFlow RSU is basically an OpenFlow switch equipped with wireless interface to communicate with the vehicles. We have implemented a component in the OpenFlow controller to automatically manage the geographical location of RSUs, which is used as a basis for our GeoBroadcast routing. In our SDN based GeoBroadcast routing, the first warning message received by source RSU is sent to the SDN controller as a packet-in message. The SDN controller will use topological and geographical information to set up the routing paths to the destination RSUs by installing appropriate flow entries on the corresponding OpenFlow RSUs and intermediate switches. Subsequent periodic messages will follow these paths to reach their destination without the controller’s intervention and certainly will reduce controller overhead, message latency, and network bandwidth consumption. Based on the proposed architecture, two use cases are realized. In addition, two application tools are implemented on top of the Floodlight controller to allow the administrators to visualize the RSUs on Google map and to broadcast emergency or warning messages.

The remainder of this paper is organized as follows. In Section II, we propose an OpenFlow-based GeoBroadcast architecture and introduce two use cases. In Section III we emulate our architecture using OpenNet. Section IV concludes our work.

II. OPENFLOW-BASED GEOBROADCAST ARCHITECTURE

A. Architecture Overview

The OpenFlow based ITS architecture is shown in Fig. 2. SDN decouples the network into control plane and data plane. In the control plane, the centralized controller uses OpenFlow protocol to control OpenFlow switches and OpenFlow RSUs. OpenFlow RSU is an OpenFlow switch, which uses the wireless interface to communicate with the neighborhood vehicles. In general, all the SDN controllers come with some common inbuilt components. We use some of these to construct two components. The first one is the switch management component and the second one is the topology management and routing component. The switch management component detects the joining and leaving events of OpenFlow switches and sends notification to other components if needed. The topology management and routing component is responsible for dynamic topology discovery, Link Layer Discovery Protocol (LLDP) is used to discover the network topology periodically and also to calculate the shortest paths for all pairs of OpenFlow switches. In order to support geographical routing, we implement two additional components for automatic RSU location management and GeoBroadcast routing in the Floodlight controller. In addition, we implement two application tools: RSU visualization tool in Google maps and emergency message broadcast tool on top of the controller to help the ITS administrators to manage the SDN-based ITS more efficiently. The detail description of these components and applications are provided in the following subsection.

Our protocol operation is shown in Fig. 3. After receiving a non-IP GeoBroadcast packet containing a GeoBroadcast header in the network layer from the source vehicle, the OpenFlow RSU looks up the flow table to search for the existing matched flow entry, installed by the OpenFlow controller. In case of a table-miss, a packet-in message corresponding to the received GeoBroadcast packet is sent to the OpenFlow controller. The application running in the OpenFlow controller decodes the packet-in message to retrieve the GeoDestination information and finds all destination RSUs within the GeoDestination area. A GeoBroadcast packet header is defined in standard [2]. It contains GeoAreaPos Latitude and Longitude which are the destination latitude and longitude. Distance $a$ and Distance $b$ are the affected range. Angle is the rotate angle of the destination region from the X-axis. Based on the GeoAreaPos Latitude, GeoAreaPos Longitude, Distance $a$, Distance $b$, and Angle, we can determine GeoDestination. Only source and destination MAC address of the vehicles is used as match field for the installed flow entries as GeoBroadcast packet is
a non-IP packet. The major differences between SDN based and traditional ITS architecture are firstly, no conversion is needed between GeoNetworking and IPv6 for SDN based approach and secondly, the SDN controller doesn’t need to process all the packets but just the packet_in message only.

B. RSU Location Management Component

In our architecture, the SDN controller uses OpenFlow protocol to communicate with RSUs which are OpenFlow switches with wireless interfaces. However, to manage the RSUs’ location, we employ an OpenFlow experimental message called the Vendor message. It provides a standard way to offer additional functionality. We define the location request and location reply messages as our Vendor messages in OpenFlow to exchange the RSUs’ location information.

The RSU location management component manages the RSUs’ location and link status information. This information is recorded in a location table. As depicted in Fig. 4 the switch management component sends a link event notification to the RSU location management component when a RSU joins the specified SDN network after completing a handshake with the OpenFlow controller. The RSU location management component sends a location request message to the corresponding RSU. We assume that the RSU is either equipped with a GPS receiver or knows its own GPS location. Thus, the RSU can insert the GPS information into the location reply message before sending it back to the controller. The component decodes the location information from the location reply message and stores the RSU’s ID, link status, and location information in the location table. Similarly, if RSU leaves a specified SDN network, our component uses the RSU’s ID to find corresponding data and deletes it from the location table in the controller. In this way an automatic management of RSUs’ location information and link status is implemented and used as a basis of our GeoBroadcast routing.

C. GeoBroadcast Component

The GeoBroadcast component running in the controller is responsible for setting up the routing paths for broadcast messages. After the source vehicle sends the first GeoBroadcast message, the nearest RSU receives it and sends a corresponding packet-in message to the controller. The GeoBroadcast component finds a set of RSUs within the GeoDestination area and redirects the message to those destination RSUs. Then, the destination RSUs receive and broadcast the message among the vehicles located within their broadcast area. The GeoBroadcast component running in the controller determines the GeoDestination area by retrieving GeoAreaPos Latitude, GeoAreaPos Longitude, Distance a, Distance b, and Angle from the GeoBroadcast packet header, which is encapsulated in the packet-in message sent from source RSU. Sequence Number in GeoBroadcast packet header can be used for distinguishing repetitive messages. The controller then finds all destination RSUs’ IDs within that GeoDestination area in the location table. The GeoBroadcast component finds a set of the shortest routing paths from the source RSU to all destination RSUs with the help from the topology manager and routing components. Finally the controller sends the flow modify messages to install routing rules in corresponding switches and RSUs for each routing path.

As the OpenFlow switch cannot identify a GeoBroadcast packet, and consider it as a normal Ethernet packet, we use the source and destination MAC address to distinguish the received packet. In the match field of a flow entry we use the vehicle’s MAC address as the source MAC and the broadcast address as the destination MAC. When an OpenFlow switch has multiple rules added, the first rule will be overwritten by the second rule if the rule uses the same match field. In order to avoid routing rules being overwritten for the multiple routing paths, sharing the same switches or RSUs, we further merge routing paths for every switch and RSU. For example, if two routing paths produced by the same GeoBroadcast message share the same switch, we assume one route’s output port is 1 and the other route’s output port is 2 respectively. If we send flow modify messages for both routing paths, then one of the routing rules in the shared switch will be overwritten. However, after the controller merges the output ports, the next switch output ports become both 1 and 2 which implies any message routed to port 1 will also copy to port 2 or vice versa. The flow entry of the switch as mentioned in above example is shown in Table 1, in the match field, the source MAC address is specified as the MAC address of the source vehicle and the destination MAC address is specified as the broadcast address. The action field is filled with multiple actions, which are output to port 1 and port 2. Subsequent warning messages will follow routing rules to reach all destination RSUs and the destination RSUs farther broadcasts them to neighborhood vehicles.

D. Use Cases

In this subsection, we consider two use cases, including static event, and moving event. We discuss issues such as RSUs transmission range overlap. Some possible solutions to incorporate packet input port numbers and hard time-out feature into flow entry installation are proposed to guarantee the robustness of GeoBroadcast function. The detailed description of each use case is as follows.

1) Static event scenario

The example of a static event is just like a car breaking down or having an accident as depicted in Fig. 5. The victim will immediately transmit periodic warning messages to notify the neighborhood vehicles. The nearest RSU, such as

<table>
<thead>
<tr>
<th>TABLE I. SWITCH’S FLOW ENTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match field</td>
</tr>
<tr>
<td>src_mac= vehicle_mac</td>
</tr>
<tr>
<td>dst_mac= broadcast</td>
</tr>
</tbody>
</table>

Fig. 4. Operation process of RSUs joins SDN.
RSU 2 in Fig. 5 will receive this message and a table-miss will occur as no existing matching rule in the flow table, resulting in a packet-in message to the controller.

![Fig. 5. Static event scenario](image.png)

The controller determines GeoDestination from the received packet-in message and searches the location table to locate RSU 1, RSU 2, and RSU 3 within GeoDestination and computes the routing paths from RSU 2 to RSU 1 and RSU 3 as example in Fig. 5. After successful installation of appropriate flow entries in the flow tables of RSU 1, RSU 2, RSU 3 and switch 1, after receiving the warning message from the victim vehicle, RSU 2 will transmit a warning message to switch 1 and also broadcast it locally. Switch 1 will route the received warning message from RSU 2 to both RSU 1 and RSU 3. After receiving this warning message both RSU 1 and RSU 3 will also broadcast it locally through the wireless interface. Subsequent periodic warning messages received at RSU 2 can follow the same routing paths, providing periodic warning messages broadcast in GeoDestination without the controller’s intervention and reducing controller overhead.

When the source vehicle is well within the transmission range of two RSUs’, both RSUs send packet-in messages to the controller. The controller will receive two packet-in messages for the same warning message. If the controller processes each of them separately and adds flow entries, the packet will be copied into two. In order to avoid duplicate packet transmission, the controller uses the GeoBroadcast packet’s sequence number to identify duplicated packets. The controller just processes one packet and adds a drop rule to one of the RSUs’ flow table, avoiding wastage of network bandwidth.

![Fig. 6. Moving event scenario](image.png)

2) Moving event scenario

In a moving event scenario, GeoDestination will change as the source vehicle moves. In this paper, we assume that a “make way for ambulance” warning is our moving event scenario. In this condition the targeted region is always the forward direction of the ambulance. In this work, we encounter two specific problems for moving event scenario, resulting in a drop in message delivery ratio as the messages failed to forward to GeoDestination.

Firstly, relying on only the vehicle’s MAC address as a match field for flow entries may introduce incorrect matching for warning messages as the source RSU and GeoDestination keeps changing in a moving event. In the ambulance scenario, the new source RSU could be the former destination RSU as an ambulance moving forward. If the controller installs flow entries with only MAC addresses as matching fields, the MAC addresses of the periodic GeoBroadcast message from the moving ambulance to the new source RSU is same as the previous, the existing flow entry will mask the generation of packet-in and force it to follow the same routing path resulting in the failure of message forwarding to new GeoDestination. Therefore, we combine the message input port with the vehicle’s MAC address to update the match field for flow entries. An example is shown in Fig. 6. At some point of time t1, RSU 1 is nearest to the moving ambulance and receives the first warning message and generates a packet-in to the controller. The RSU 1, RSU 2 and RSU 3 are in GeoDestination area. Controller decouples packet-in, computes routing paths, and installs flow entries in switch 1, RSU 1, RSU 2 and RSU 3 with MAC address and input port as a match field. For example, match field’s input port of RSU 2 is from Switch 1. Later on as the ambulance moves forward the GeoDestination changes, in time t2, RSU 2 becomes the nearest to the moving ambulance and receives the periodic GeoDestination message form the wireless input port. RSU 2 looks up the flow table and generates a packet-in to the controller again as the message input port is different from previously installed rule’s input port (from switch 1). Similarly, the controller installs flow entries in the switch and RSUs to allow subsequent messages to follow routing paths and to reach GeoDestination successfully.

![Fig. 7. Calculation of hard time-out](image.png)

Secondly, as the moving GeoDestination covers new RSUs or existing RSU may leave the moving GeoDestination, the RSU’s old routing rule is still active and may also introduces incorrect matching for warning messages. Fig. 7 illustrates the problem. At time t1, RSU 1 receives the first warning message and generates a packet-in to the controller. The controller sets the forwarding rules to cover RSU 1, RSU 2 and RSU 3 within GeoDestination. However, at time t2 the ambulance moves to a new location with GeoDestination covering the new RSU (RSU 3). The old rules set in t1 cannot forward a warning message to RSU 3. Furthermore, at time t3 RSU 3 is no longer in GeoDestination. The old rules will force RSU 1 to broadcast a warning message continually. Therefore, we install each flow entry with a hard time-out to solve this problem. As per OpenFlow standards, hard time-out means the flow entry will be removed after a specified time in second. Hard time-out calculation is the key to this solution. There are two timing points to time-out old rules. The first
one is, the RSU\textsubscript{src} is going outside the moving GeoDestination, and another one is new RSUs are going to covered by the moving GeoDestination. Since the warning message contains the vehicle’s location, moving direction, and speed information, the controller can use this information and RSU\textsubscript{src}’s location to predict the elapsed time when RSU\textsubscript{src} is most likely to be outside of GeoDestination. We calculate hard Time-out using the distance between the source vehicle and source RSU divided by vehicle speed as shown in equation (1).

\[ \text{Timeout}_{\text{RSU\textsubscript{src}}} = \frac{\text{distance(\text{vehicle}, \text{RSU\textsubscript{src}})}}{\text{speed\textsubscript{vehicle}}} \] (1)

The second timing point happens when moving GeoDestination covers a new RSU. We calculate the distance between source vehicle and a new covered RSU. The distance minus the size of GeoDestination gives us how far the GeoDestination will move forward to cover a new RSU. We divide it by the vehicle’s speed to get the Time-out value as shown in equation (2).

\[ \text{Timeout}_{\text{cover new RSU}} = (\frac{\text{distance(\text{vehicle}, \text{RSU\textsubscript{new covered}})}}{\text{size GeoDestination}}) - \frac{\text{size\textsubscript{GeoDestination}}}{\text{speed\textsubscript{vehicle}}} \] (2)

As shown in Fig. 7, at time t1, RSU\textsubscript{1} receives the first warning message and generates a packet-in to the controller. Controller calculates that the moving GeoDestination will cover a new RSU (RSU\textsubscript{3}) after travelling a distance of 100m. The controller uses vehicle location, speed, RSU location, and size of GeoDestination to calculate the hard time-out using Eq. (2), which is 100m divided by vehicle speed. Then the controller installs the flow entries in corresponding RSUs. At time t2, the old rules in t1 are time-out. Therefore, the periodic warning message received by RSU\textsubscript{1} could not find a match and will generate a packet-in to the controller. Controller predicts that the moving GeoDestination will leave the existing RSU (RSU\textsubscript{1}) after travelling a distance of 100m. Using Eq. (1), the controller calculates the hard time-out again and installs new flow entries. At time t3, the old rules installed in t2 are time-out. The controller will install new flow entries for next packet-in message.

When the vehicle is well within the transmission range of two RSUs, a moving event must be specially considered even though in a static event a vehicle’s message can be easily identified as a duplicate by the message’s sequence number. For example, in a moving event the ambulance is in the overlapping transmission range of RSU\textsubscript{3} and RSU\textsubscript{4} as shown in Fig. 6. We assume both RSUs are in GeoDestination. RSU\textsubscript{3} first receives a “make way” warning message from the vehicle and generates a packet-in to the controller. The controller installs a flow entry to corresponding RSUs including RSU\textsubscript{2}. Since we assume the warning message will be sent to RSU\textsubscript{3} from RSU\textsubscript{2} through Switch1, the matching field will include not only the source vehicle MAC address and broadcast MAC address, but also the message’s input port (Switch1). The controller stores the RSU\textsubscript{3} ID and uses the sequence number to make sure another warning message packet-in by RSU\textsubscript{2} is duplicated. Then the second warning message received by RSU\textsubscript{3} will follow the rule installed by the first warning message. However, the second warning message received by RSU\textsubscript{2} is from the wireless interface rather than from the Switch1 physical port. It will mismatch the matching field containing the MAC addresses and the Switch1 physical input port installed by the first warning message. Thus, RSU\textsubscript{2} generates another packet-in with the second sequence number because the message input port now is different. Therefore, the message cannot be identified as a duplicate by the controller. The controller computes the routing paths and installs appropriate flow entries again to the corresponding RSUs, resulting in duplicate flow entries in the RSUs for the same warning messages. To solve this problem, we need to drop one of the warning messages. When the controller discovers that the source vehicle is well within the transmission range of two RSUs, the controller installs a dropping rule in one of the RSUs. From the driving direction of the source vehicle, we can decide which RSU will be moving away from the source vehicle gradually. We install drop rules in that RSU to avoid the message being copied into two. It can further reduce network bandwidth consumption.

E. Application

Finally, we use the above RSU location management and GeoBroadcast components to implement RSU virtualization and emergency notification APPs. The Floodlight controller uses Rest API as its Northbound API. In the Rest API, developers can customize application requests to access component information. We implement a RSU virtualization application. Users can send requests to our map server and the server will get RSUs’ information through Rest API. In addition, users can show RSUs’ geographic location on Google map as shown in Fig. 8. For emergency notification APP, administrators can decide a GeoDestination and sends a GeoBroadcast warning message request to APP, afterwards the controller will send GeoBroadcast packets to all RSUs within that GeoDestination. Finally, RSUs broadcast the received packets to vehicles, to deliver warning messages.

III. PERFORMANCE EVALUATION

We use OpenNet [8] to simulate our design which combines the mininet and ns-3 wireless modules to simulate vehicle and RSU and implement an application in Floodlight [9] to automatically manage the geographical location of RSUs. The emulation parameters are shown in Table II. The source vehicle transmits four messages per second. In static scenario, GeoDestination is a circle, so we change GeoDestination range in radius. The packet delivery ratio is as shown in Fig. 9(a). Delivery ratio is similar as a whole. As per Fig. 9(b), controller overhead in traditional ITS architecture is much higher than in SDN. Fig. 9(c) shows that the SDN bandwidth consumption is in between 32–41% of that of ITS. In a moving ambulance scenario, GeoDestination
is a rectangle with fixed width and variable length. In delivery ratio, SDN and ITS are almost the same except in a range of 500m as shown in Fig. 10(a). We can observe that the SDN controller’s overhead is reduced to about 2~8% of that of ITS in Fig. 10(b). In addition, SDN bandwidth consumption is reduced to 20~40% of ITS in Fig. 10(c). The effect of RSU’s transmission range overlap issue in moving events can be observed in Fig. 11(a). SDN’s delivery ratio is 3~5% lower than that of ITS. The main reason is the precision of hard time-out is not very accurate. As the hard time-out precision in OpenFlow switch is limited to one second, the new flow entries can’t be added in accurate time. In this experiment, we found that there are about 31~40 times of hard time-out happen. SDN controller overhead is 7~11% of ITS in Fig. 11(b) and SDN bandwidth consumption is 33~38% that of ITS in Fig. 11(c).

IV. CONCLUSION

We propose a Software Defined Network (SDN) architecture for GeoBroadcast in VANETs. The source vehicle sends periodic warning messages to notify destination vehicles. We implement RSU location management and GeoBroadcast components in the Floodlight SDN controller that will control OpenFlow RSUs and switches to support efficient transmission and delivery of periodic warning messages to the destination area. Our SDN based design overcomes the drawback of the existing ITS which requires a conversion of warning messages between GeoNetworking and IPv6. Furthermore, we achieved a significant reduction of 84% in controller overhead and a reduction of 60% in network bandwidth consumption. References