Improvement of Security and Scalability for IoT Network Using SD-VPN

Linda Shi†, Fei Wang‡ and Chung-Horng Lung‡
lshi030@uottawa.ca, feiwang6@cmail.carleton.ca, chlung@sce.carleton.ca
†School of Electrical Engineering and Computer Science, University of Ottawa, Canada
‡Department of Systems and Computer Engineering, Carleton University, Canada

Abstract— **The growing interest in the smart device/home/city has resulted in increasing popularity of Internet of Things (IoT) deployment. However, due to the open and heterogeneous nature of IoT networks, there are various challenges to deploy an IoT network, among which security and scalability are the top two to be addressed. To improve the security and scalability for IoT networks, we propose a Software-Defined Virtual Private Network (SD-VPN) solution, in which each IoT application is allocated with its own overlay VPN. The VPN tunnels used in this paper are VxLAN based tunnels and we propose to use the SDN controller to push the flow table of each VPN to the related OpenvSwitch via the OpenFlow protocol. The SD-VPN solution can improve the security of an IoT network by separating the VPN traffic and utilizing service chaining. Meanwhile, it also improves the scalability by its overlay VPN nature and the VxLAN technology.**

***Index Terms*— SDN, IoT, SD-VPN, OpenFlow, VxLAN, overlay VPN, service chaining**

I. INTRODUCTION

Internet of Things (IoT) can be applied to a wide range of areas, such as connected industry, smart city, smart energy, connected cars, etc. Almost all of these applications require security. For example, no one wants the autopilot car to be hacked when that the car is in use and no enterprise can afford the consequence of data leakage. However, in an IoT world, since everything can be connected to the Internet, which means any IoT system is exposed to the Internet and there is a high risk to be attacked. Hence, security is one of the top priorities. There are various proposed approaches, such as [1], [2] to address the IoT security issue. However, most of them are from cryptography and encryption algorithmic perspective and none of them solves the problem from the network aspect, or to be more specific, from the network separation's point of view. In this paper, we introduce a new type of network solution for IoT. In our solution, we propose to build different Virtual Private Networks (VPNs) for different IoT applications using the Software-Defined Networking (SDN) technology. With the approach, each application has its own private VPN network, it is logically separated from other networks and the Internet. We can also utilize SDN to push advanced security policy to the OpenFlow-enabled network nodes which are at the network edge to provide network connections to the IoT devices.

As stated previously, with the development of IoT, anything can be connected to Internet. The Internet of objects would encode 50 to 100 trillion objects, and be able to follow the movement of those objects. Human beings in surveyed urban environments are each surrounded by 1000 to 5000 trackable objects [3]. And each object needs a unique address. This will consume 50 to 100 trillion IP addresses and the routing tables on the core routers can have the explosion problem. As a result, scalability is another main challenge. To address this issue, a popular method is to deploy IPv6 [4][5]. But IPv6 requires both hardware and software upgrade and new routing protocols also need to be implemented to support IPv6. With our proposed SD-VPN solution, each IoT application is allocated with its own an overlay VPN. Consequently, each IoT application can use overlapping private IP addresses, as long as the service provider's underlay IP address space is public and unique. As for the size of routing/switching tables, core routers only hold the routing/switching table of the underlay network which is the core network of the service provider since the overlay packets are encapsulated as payload in the VPN tunnel.

We have evaluated the proposed solution using a testbed built with the OpenDaylight Boron SR3[6] as the SDN controller and Raspberry Pi devices as Openflow switches .

II. SD-VPN FOR IOT NETWORK

A. SD-VPN

Currently, the majority of VPN services are provisioned manually. Consequently, it may take a few days to build the VPN services, as a variety of configurations need to be changed. The manual process delays new service implementation and increases operational cost and complexity. As we all know, SDN provides network simplification and automation, it is effective to provision the VPN services via SDN and this is so-called SD-VPN. In other words, instead of creating VPN services manually in advance, SD-VPN creates VPN services automatically when a VPN client joins.

The basic working principle of SD-VPN is shown in Fig. 1. PE (Provider Edge) routers, commonly used in VPN services, are replaced by SDN switches (SD-VPN endpoint as shown in Fig. 4) which are controlled by the SDN controller via OpenFlow. As a result, PE to PE control plane protocols such as MP-BGP are eliminated [7]. The SD-VPN endpoints could be an enterprise branch office gateway router, a home gateway,

978-1-5386-3416-5/18/$31.00©2018 Crown

a LTE eNodeB or any OpenFlow physical or virtual switch. Initially, there is no VPN service on the SD-VPN endpoints. When the first VPN client (e.g., a laptop, a smart phone, an appliance, or a vehicle, etc.) is powered on and joins its own VPN, the client is booted up with its unique metadata (e.g., MAC address, DHCP option 82, port number, etc.) which distinguishes itself from other VPN services. The mapping of metadata to different VPNs, in other words, to different IoT services is pre-defined in the cloud (e.g., could be on the controller or on the orchestration platform). The SD-VPN endpoint attached to the client detects the event and notifies the SDN controller. The controller then further queries the application/management layer which VPN instance should be allocated to this client. Based on the metadata mapping configuration provisioned in the application or management platform, the controller gets the network information from the application/management layer, including the VPN instance, IP address, security and quality of service (QoS) policies. Then the controller pushes that information to the SD-VPN endpoint and a VPN instance is created on the endpoint accordingly. When the second or any following client belonging to the same VPN is booted, the same procedure is repeated and the VPN instance and the VPN tunnels are pushed to the SD-VPN endpoints. In this paper, we only discuss VxLAN VPN tunnels.

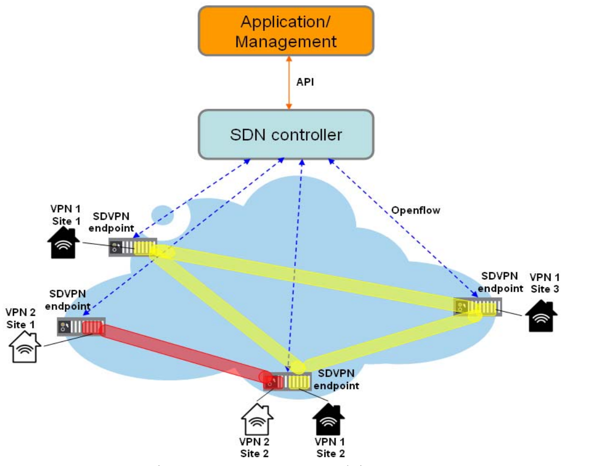

Fig.1 SD-VPN Architecture

B. *Architecture and Deployment of SD-VPN for IoT Network*

As discussed earlier, SD-VPN can provide VPN services automatically. In other words, SD-VPN can establish VPN tunnels and connect end points into their own VPNs automatically. In addition, the routing table of the overlay VPN service is built by the SDN controller at the same time when the VPN is created, and the controller can push the routing table to every involved router or switch via OpenFlow. The above features make SD-VPN suitable for IoT networks.

Fig.2 illustrates our proposed solution of SD-VPN deployment for IoT networks. There are three key components in our solution: orchestration platform, SDN controllers and the OVS on the network edge devices. Since the VPNs are overlay network instances, our proposed solution is agnostic about the underlay infrastructure, which means the proposed solution can be deployed in service providers' core networks, e.g., IPv4, IPv6, MPLS. The primary function of underlay network is to provide IP connectivity between the orchestration platform, SDN controller and the network edge devices, e.g., smart home gateways, PE routers, eNodeBs or smart phones, which are installed with OVS (OpenvSwitch).

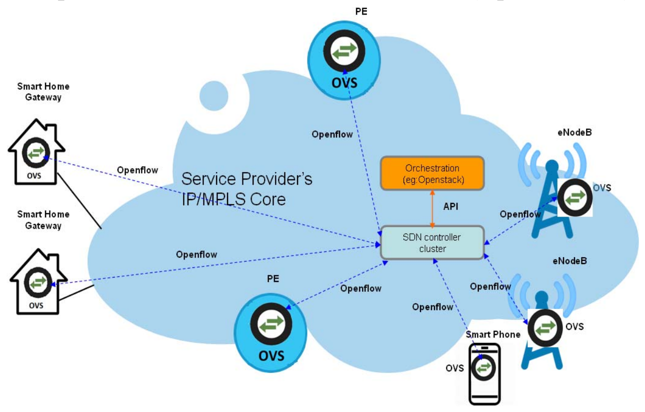

Fig.2 SD-VPN Deployment for IoT Network

Firstly, the orchestration platform is to provision the overlay service and the security and QoS policies. The orchestration platform could be OpenStack [8], CloudStack [9] or any cloud management system. With the orchestration platform, we can have centralized and unified provisioning throughout the network. As OpenStack and CloudStack have been widely deployed, advanced and complex policies are possible. Another benefit of having the orchestration platform is that it is open source and provides southbound API to the SDN controller, which makes our proposed solution flexible and programmable.

Secondly, the SDN controller can be implemented as a dedicated or a virtual server. Its north-bound API towards the orchestration layer is to get the overlay network service template and policies, while the south-bound OpenFlow interface is to push the overlay network configuration and flow tables to the OpenvSwtich on the edge devices.

Lastly, the home gateways, PE routers, eNodeBs or smart phones should be OpenFlow-capable, since they are managed by the controller via OpenFlow. If possible, we can upgrade the existing edge devices by installing the OVS module. Same as the controller, those edge nodes can be either physical or virtual with Network Function Virtualization (NFV) [10], which eliminates the hardware requirement. Later, the overlay VPN services will be created on top of the OVS and the forwarding and encapsulation table of each VPN instance will be pushed to the OVS from the controller via OpenFlow.

Our proposed solution can be applied to various types of IoT applications, such as remote control of smart appliances, remote control of home temperature, smart TV/mobile TV connection to content delivery networks (CDNs), autopilot of smart cars, and so on. Take autopilot of smart cars as an example. With our solution, road side units RSUs [11] are not needed, as eNodeBs or base stations can be used to connect smart cars as long as they support OpenFlow. This can save a considerable amount of cost of deployment RSUs, since eNodeBs and base stations are already available. To guarantee the security of the vehicular network, a dedicated VPN can be created for all the smart cars, traffic lights and the intelligent transportation system (ITS).

C. Overlay VPN Network

The overlay VxLAN VPN services for IoT applications proposed in this paper have the following characteristics:

(1) Scalable. As shown in the logical overlay topology in Fig. 3, there is one overlay VxLAN VPN per IoT application, so that all different applications are logically separated from each other. VxLAN can support 16 million VNI, which means that it allows up to 16M VxLAN VPNs to coexist within the same administrative domain. We will further discuss the enhanced scalability in the next section.

(2) Dynamic. The overlay VxLAN VPNs and VxLAN tunnels are created if needed. In other words, there is no overlay VPNs on the OpenvSwitch on the edge nodes initially. When a client starts, e.g., a user turns on the smart TV, a user starts a smart car or launches an App on the smart phone, a corresponding overlay VPN is created on the OpenvSwitch of the edge node connecting this IoT device/App.

(3) Automatic. The overlay VxLAN VPNs and VxLAN tunnels are created automatically by the controller, which reduces the operational cost and complexity.

(4) Manageable. The centralized SDN controller makes the proposed solution easily manageable. Also, the controller has a programmable northbound API, which gives more granular capabilities for configuration and management.

D. Control Plane Message Flow

This section discusses the control plane signaling procedure when an overlay VxLAN VPN is created. Fig. 3 is an example of control plane message flow of a simple Smart Temperature service which connects the temperature setting panel at home and the Temperature App installed on smart phones which allows the user to remotely control the home temperature and the local Hydro network for energy monitoring or saving. The detailed steps are described as follows:

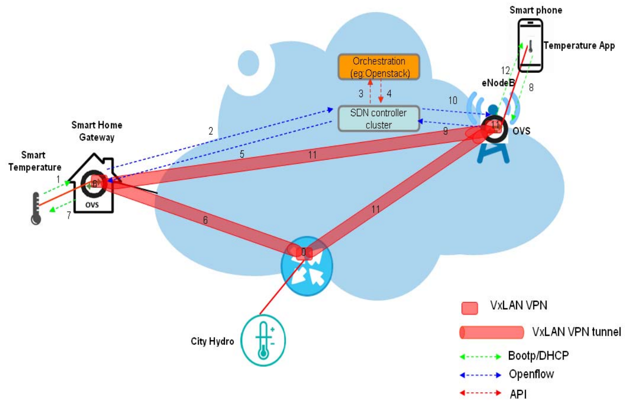

Fig.3 Control Plane Message Flow

Step 1: The smart temperature control panel is initialized at home. A DHCP Discover message is broadcast and detected by the OVS on the smart home gateway.

Step 2: The OVS reports this new device activation to the SDN controller in the service provider's network via OpenFlow. There is metadata associated with each IoT device/App to distinguish itself from others, and the metadata is also included in the OpenFlow message.

Step 3: The controller checks its database and there is no entry for this service, so the controller queries the orchestration layer via the north-bound API.

Step 4: Based on the metadata, the orchestration layer selects the corresponding overlay network and its associated QoS and security policies for this IoT device/App. Then the orchestration layer responds to the controller with these selected parameters (VNI, IP address, policies and etc.).

Step 5: In turn, the controller replies to the OVS with the overlay parameters and flow table in an OpenFlow message.

Step 6: Once the OVS gets the configuration parameters, an overlay VPN instance is created on the OVS with related VxLAN tunnel (to the PE facing the network).

Step 7: The OVS sends a DHCP OFFER message to the temperature control panel. Now the control panel is assigned with its IP address and connected into its own VPN.

Step 8: Assuming that the home owner is outside and wants to control his home temperature remotely from his smart home. The user launches a Temperature App and this App is treated as a virtual IoT end device and the first step is to request an IP address via the DHCP.

Step 9: The OVS on the eNodeB or base station connecting to the smart phone receives the DHCP Discover message and translates it to OpenFlow and send it to the SDN controller.

Step 10: The controller checks its database and this time there is already an entry for this service, i.e., the Smart Temperature service, so the controller replies to the OVS with the corresponding overlay parameters and flow table using OpenFlow immediately without querying orchestration layer.

Step 11: Same as step 6, an overlay VPN instance is created at the OVS on the eNodeB with related VxLAN tunnels (to the PE facing City Hydro and the home gateway).

Step 12: A DHCP OFFER message is sent to the App on the smart phone and the App is connected to the temperature VPN and it can control the home temperature remotely.

E. Data Plane Encapsulation

This section describes how the data plane packets traverse the network.

Fig. 4 illustrates the encapsulation and de-capsulation of packets between the temperature setting panel at home and the Temperature App installed on smart phones. Assume the MAC address of the temperature setting panel is MAC11 and the Temperature App is MAC12. The IoT devices have no idea about the underlay network, so they exchange packets normally. When the packets arrive the home gateway or the eNodeB, the OVS encapsulates the packets into appropriate VxLAN tunnel with the VNI allocated to this overlay service.

VxLAN is based on UDP/IP, so the VxLAN packets need to be further encapsulated into IP packets and this IP header is called the outer header. The source and destination IP addresses in the outer header is the IP addresses of home gateway and the eNodeB. In summary, the IP addresses in the outer header are the IP addresses of the endpoints of the VxLAN tunnel. When the other end receives the packets, it first removes the IP header, then according to the VNI in the VxLAN header, it forwards the inner packets to the overlay VPN they belong to.

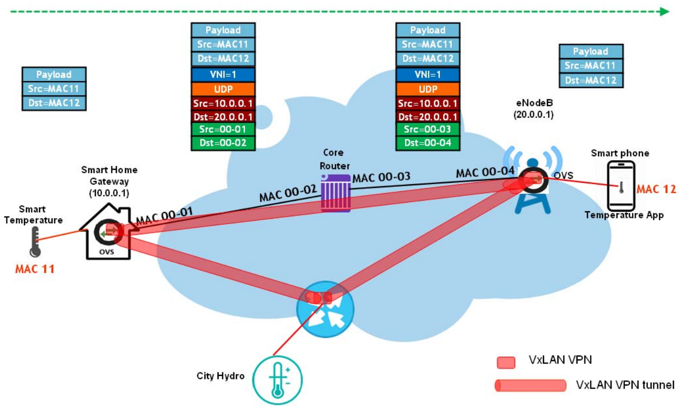

Fig.4 Data Plane Encapsulation

III. Improved Security and Scalability

A. *Improved Security*

The proposed SD-VPN combines SDN and VPN, i.e., it uses characteristics of both SDN and VPN to improve network security in two ways: (i) Traffic separation provided by VPN; and (ii) Network service chaining based on SDN.

1) Traffic separation

IoT end-point devices are allocated into different VPN sites in our SD-VPN architecture. Tunneling is the basic technology of VPN, which is similar to the point-to-point connection technology. SD-VPN separates the network traffic, which enhances security, because users can only access a unique VPN site via the tunnel. Our proposed approach can separate sensitive data with other parts of the entire network. By doing so, the proposed solution protects confidential data and prevents data leaking. In different VPN domains, datagrams are separated when devices are exchanging information. Users in the one VPN domain cannot transmit data to other users in different VPN domains.

2) Network service chaining

Network Service Chaining [12], which is also named as Service Function Chaining (SFC), is not a totally new concept. With the development of SDN technology and NFV, SFC has gradually become more important. Using SDN capabilities, SFC builds a virtual service chain of network services.

In the SD-VPN architecture, we can use SFC to enhance network security. When we need to access to a specific VPN site, the request is sent to an OpenvSwitch capable router. Before we access to the destination site, the SDN controller can set up a service chain, like firewall and attack detection, to detect the data request. Using this approach, all user traffic needs to go through security check before to accessing a VPN site. Thus, the network security can be improved.

B. *Improved Scalability*

Compared to the traditional IoT network, SD-VPN has several advantages in scalability. First, VPN introduces the overlay concept, i.e., the IP address space used in each overlay VPN instance can overlap with each other due to the encapsulation of VxLAN tunnels. Second, the routing table for core router is much smaller than the large routing table of the traditional IoT network, as the core network is only responsible for the underlay routing and forwarding. Lastly, the flow table of the edge OVS (open virtual switch) is smaller than the large flow table of the traditional IoT network.

VxLAN is a simple mechanism for encapsulating MAC in UDP and can create virtual layer 2 subnets across multiple physical IP subnets. VxLAN is a publicly known solution to mitigating the limitations (or improving the scalability) of existing 802.1Q or Q-in-Q VLANs. Our solution utilises the SDN technology which decouples the control and data planes, which in turn reduces the control overhead for switches.

Our SD-VPN solution is also scalable in control plane. Without SDN, PEs need to exchange signalling messages (eg: ospf, bgp, etc) between each other. Now with SDN, the routing protocols are no longer needed, instead openflow messages are exchanged between the controller and the OVS to maintain the forwarding table. And Openflow is much more light-weight compared to routing protocols.

IV. Conclusions

IoT has generated a great deal of interests. There are still issues in security and scalability. In this paper, we proposed a SD-VPN architecture to address these two issues from the networking perspective. Further, we have evaluated the proposed solution using real devices, i.e., Raspberry Pi, and the OpenDaylight SDN controller.

Based on the experimental results, we can conclude that using SDN technology to deploy VxLAN VPN has evident benefits, such as: (1) automated service provisioning reduces the complexity and cost of operation; (2) virtual separation of services with a centralized and in-depth security policy and service chaining can improve the security; and (3) VxLAN and the overlay model are more scalability. All the above benefits are suitable for IoT networks.

References

[1] J. Granjal, E. Monteiro, J. S. Silva, "Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues", *IEEE Communications Surveys & Tutorials*, 17 (3), 2015. ,

[2] S. Kulkarni, S. Durg, N. Iyer, "Internet of Things (IoT) Security", *Proc. of the 3rd Int'l Conf. on Computing for Sustainable Global Development*, 2016.

[3] Internet of Things, https://en.wikipedia.org/wiki/Internet_of_things #cite_note-Waldner.2C_2007-98.

[4] A. Jara, L. Ladid, D. El Ouadghiri, "Challenges of the Internet of Things: IPv6 and Network Management", *Proc. of the 8th Int'l Conf. on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014.*

[5] Teemu Savolainen, Jonne Soininen, Bilhanan Silverajan, "IPv6 Addressing Strategies for IoT", *IEEE Sensors Journal*, 13(10), , October 2013.

[6] OpenDaylight https://www.opendaylight.org/what-we-do/current-release/boron

[7] B. Mirkhanzadeh, N. Taheri, S. Khorsandi, "SD-VPN: A Software-Defined Solution for VPN Service Providers", *Proc. of IEEE/IFIP Network Operations and Management Symposium*, 2016.

[8] Openstack, https://www.openstack.org/

[9] Cloudstack https://cloudstack.apache.org/

[10] Network Function Virtualization, https://en.wikipedia.org/wiki/ Network_function_virtualization

[11] B. B. Dubey, N. Chauhan, N. Chand, "Efficient data scheduling technique at RSU for vehicular ad-hoc networks", *Proc. of Int'l Conf. on Information Communication and Embedded Systems (ICICES)*, 2016.

[12] Network Service Chaining, https://www.sdxcentral.com/sdn/network-virtualization/definitions/what-is-network-service-chaining.