Virtualized and Self-configurable Utility Communications Enabled by Software-Defined Networks

Young-Jin Kim
Bell-Labs, Alcatel-Lucent
young.jin_kim@alcatel-lucent.com

Keqiang He
University of Wisconsin
keqhe@cs.wisc.edu

Marina Thottan
Bell-Labs, Alcatel-Lucent
marina.thottan@alcatel-lucent.com

Jayant G. Deshpande
Bell-Labs, Alcatel-Lucent
jayant.deshpande@alcatel-lucent.com

***Abstract* — Utility communications are increasingly required to support machine-to-machine (M2M) communications for hundreds to millions of end devices ranging from meters and PMUs to tiny sensors, high-powered sensors (*e.g.*, intelligent electric devices), and electric vehicles. The Software Defined Network (SDN) concept provides inherent features to support in a self-configurable and scalable manner the deployment and management of existing and envisioned utility communication networks that will connect between end devices and application servers, or among end devices.**

The programmability of SDN technology allows the agile, elastic, and scalable deployment of present and future utility applications with varying requirements on security and time criticality. In this work, we first show that a well-known standard solution (*i.e.*, IEEE 802.1Q [1]), which is popularly employed for virtual networking in industry, is limited to support large-scale utility M2M applications. Next, with some utility application use cases, we demonstrate that using the SDN technology (*i.e.*, OpenFlow [2]), we enable elastically adaptable virtual utility network slices per-application to securely, dynamically, and cost-efficiently meet the utility communication needs. Specifically, we design a SDN-based architectural solution for virtual utility networks that will support self-configurable, secure, and scalable deployment of utility applications that leverage many end devices. Using two SDN-enabled Ethernet switches [3] available in today's market, the feasibility of our idea is discussed.

I. INTRODUCTION

With Smart Grid roll-out, M2M communication networks supporting electric utility applications traffic is undergoing a tremendous change both in the increasing number of new utility (or grid)[1] applications, and a massive number of communication endpoints that the network must support [4]. Most of this increase in endpoints comes from deployment of sensors, currently limited to a few hundred Remote Terminal Units (RTUs), to hundreds to several million sensors including meters, Phasor Measurement Units (PMUs), Intelligent Electronic Devices (IEDs), and sensors attached to Electric Vehicles (EVs) and Distributed Energy Sources (DERs). Not only a massive number of sensors are being deployed, but also the combination of the number of sensors and their respective frequency of reporting the measurement data and status update will result in increased network traffic, as shown in Fig. 1. In addition, due to the scale, the grid applications require self-configurable and scalable communication networks that can elastically meet the needs for availability, security, and performance.

In this work, we show that the SDN technology [2] can be used as a key solution that addresses the above requirements for the fast deployment of grid applications (hereafter, called utility M2M applications), as shown in Fig.2. Specifically, we design a SDN-based architectural solution for virtual utility networks, *SVUN,* which will support in a self-configurable, secure, and

[1] We refer to grid and utility inter-changeably throughout this paper.

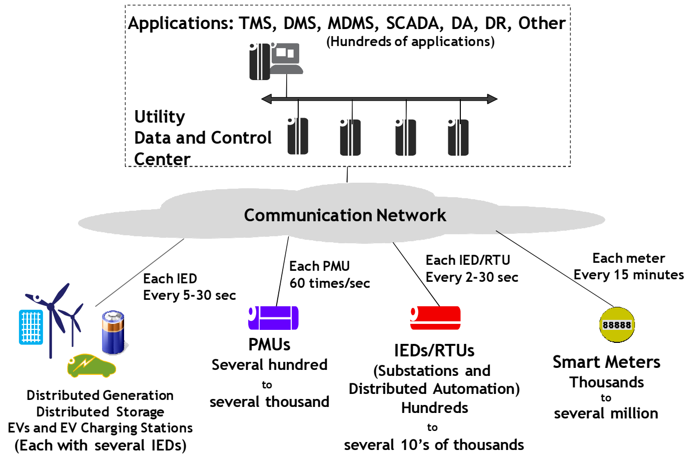


Figure 1 Smart Grid M2M Applications: Sensors, Application Servers and Communication frequency.

scalable manner the deployment of a broad spectrum of utility M2M applications in practice. Our proposed SVUN idea, which uses commoditized SDN switches as network elements, not only has an ability to in a self-configurable manner define virtual network slices with each slice that supports application (in one utility or across more than one multiple utilities), or a group of similar applications, but also will provide secure and cost-efficient communications for utility M2M applications. Further, while this paper is mostly dedicated to a SDN-based network solution for utility M2M applications, our proposed idea can be extended to a more general solution that supports other M2M applications beyond Smart Grid context.

The paper outline is as follows: In Section II, we describe challenges of today's virtual communication technology that is popularly used in industry. Overall benefits of using SDN for utility M2M applications are discussed in Section III. In Section IV, our proposed idea *SVUN* is in a detail presented. In Section V, we discuss the feasibility study performed over a small-but-real SDN test-bed which consists of SDN-enabled switches and Linux PCs. Conclusions are shown in Section VI.

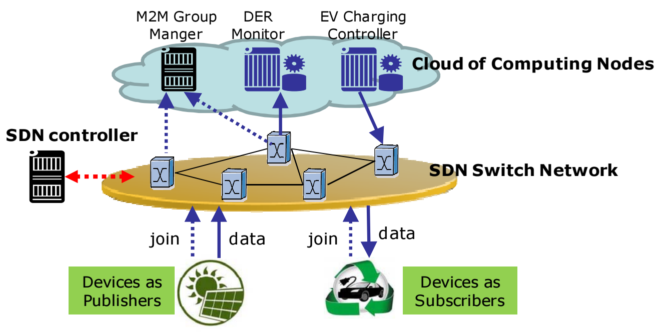


Figure 2 An Instance of SDN-enable Utility Comm. Networks.

II. Today's Virtual Communication Networks

M2M communication pattern is at the heart of the utility applications. The communication pattern observed in most of the utility applications (including the ones described in Section I) is indeed M2M with little or no human intervention. We can consider the industry standard (*i.e.,* IEEE 802.1Q Virtual LAN [1]) for virtual networking as a M2M communication solution that can accommodate the grid applications described already in Section I. Specifically, we can imagine Virtual Utility Networks (VUNs) that are deployed in diverse granularity: per-utility, per-application in a utility, per-organization in a utility, and per-application across multiple applications. However, we find out that the use of such virtual network technology incurs some limitations, as will be describe next.

Operation Complexity: When communication-enabled devices such as AMI meters, PMUs, IEDs, EVs, DERs, and so on are plugged-in into utility communication networks and need to be configured, it is highly desirable that the configuration of connectivity, security policy, Quality of Service (QoS) policy, and so on is fully automated with no human intervention (*i.e.,* self-configuration) for addressing cost-efficient and agile utility communication network operations. Otherwise, due to the scaling of the end devices and the heterogonous requirements of grid applications, utilities will face the increased management complexity along with the deployment of Smart Grid and as a result have to fairly increase the operation cost. Further, the self-configuration requirement necessities for fast restoration from failures caused by malfunctions or cyber attacks. However, we question whether conventional networking solutions including IEEE 802.1Q can address the self-configuration requirement for the utility communications described above since they provide, to the best of knowledge, manual configuration methods.

Scaling Issue: We have so far described challenges on aspects of network or system operations. We now discuss scaling issues when utility M2M communications are built over a conventional VLAN technology, *i.e.,* IEEE 802.1Q.

Consider the following deployment scenario of a million scale communication-enabled measurement and monitoring end devices; a relatively-small number (e.g., 100~1000) of network switches; thus, a physical port (called port) in a switch must be logically (not physically) connected to more than one end device (i.e., multiple meters per port via a data concentrator); also, an end device must subscribe to more than one VLAN since it can communicate with multiple different application servers, *e.g.,* data measured from a residential meter is concurrently sent to multiple different machines in a utility control center such as load prediction servers and billing servers. Unfortunately, the VLAN standard, IEEE 802.1Q, cannot scalably support the scenario due to the small number of VLANs per-port (Port-based VLANs or Protocol-based VLANs). In the port-based VLAN, an access port between a switch and access devices (not a trunk port between switches) are assigned to a VLAN during a certain time period. In the protocol-based VLAN, one VLAN per protocol is supported. As a result, an access port must be concurrently used by multiple VLANs and only a small number of well-known protocols (*i.e.,* IP, ARP, and IPX) are supported.

Security Perspective: In a VUN built with the IEEE 802.1Q, a member that are once authenticated can communicate with all authenticated members in bi-directional directions. Unfortunately, we cannot prevent end M2M devices from being compromised due to the property of M2M communication [5]. Thus, compromising even an end M2M device can result in the propagation of security threats across the VUN because of the limitations of IEEE 802.1X [6] that IEEE 802.1Q uses as a default authentication measure. Additionally, IEEE 802.1X does not supports fine-grained (application-level) granularity but only port-level granularity.

III. Benefits of SDN for Utility M2M Communications

All SDNs can provide isolation of different traffic types, applications, and/or endpoint classification, *e.g.,* virtual network slices may be defined for AMI, SCADA, DG/DS/EV, and PMU traffic. Virtual network slices may also be based on geographical or domain considerations (transmission and distribution or security zones). The virtual network slices inherently enhance security with traffic isolation and enabling security, QoS, and even network management policies for each network slice. Thus, a closed group of applications/application type/endpoint group can have its "own virtual network" that is its network slice. Note that the ability to rapidly create required functions with few changes in the physical network makes the utility network less vulnerable to potential network failures.

Our SDN-based design offers a programmable open interface to the applications as well as to the network elements for their control, configuration, and management. There is a deliberate shift from fixed network functions serving many applications to per application virtualized functions making introduction of new applications as well as connecting new endpoints in the network more efficient and manageable. The ability to reconfigure a software defined network and rapidly deploy virtualized network functions allows for greater network utilization, global resource optimization, and enhanced scaling. Thus, demand driven service and device activation and provisioning will directly lead to dynamic application velocity and scale.

The traffic isolation function of SDNs can be also useful to service providers such as Verizon who provide communication services for multiple different utilities (*i.e.,* multi-tenancy). A service provider can then create one or more network slices to support a utility's needs for (groups of) applications over these virtual network slices. All advantages of rapid creation of virtualized network functions can still be available to the service provider and even to the utility.

IV. Our Proposal: SDN-enabled Virtual Utility Network

A. Background and Assumptions

To support large-scale M2M communications, our design has been inspired by the SDN concept (majorly described in this paper) and the publisher-subscribe (Pub-Sub) communication paradigm [7]. A utility application such as smart metering that consists of homogenous measurement devices as publishers and utility head-ends as subscribers in a utility control center can be represented as a Pub-Sub group. For the past two decades, the Pub-Sub paradigm has been popularly applied to commercial

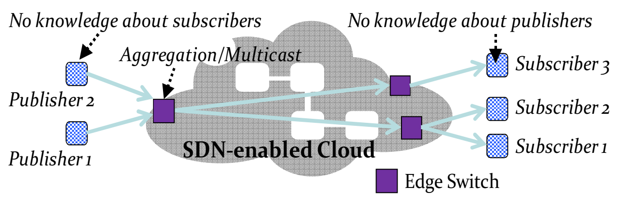


Figure 3 Publish-Subscribe Communication Paradigm.

message-oriented middleware and Internet web applications. Its distinguishing property is that each member in a Pub-Sub group does not require specific knowledge (*i.e.,* IP address) about other members in the group, *i.e., the space/time decoupling property*. In addition, information multicast/aggregation capabilities are inherently provided by the Pub-Sub system, as shown in Fig. 3. Due to the space/time decoupling property and the inherent multicast ability, the Pub-Sub paradigm has resilience against cyber-attacks and internal system failures and in turn avoids single points of failure and bottlenecks. In addition, it can support communication flows for a massive number of end devices in a scalable fashion. Thus, the Pub-Sub paradigm can be considered as a base block for utility M2M applications.

However, the Pub-Sub paradigm is implemented, to the best of our knowledge, only on overlay networks with none or very limited interfaces with physical switches and equipments. This is because most current physical switches are limited to support the pub-sub paradigm described above. Consequently, current industry Pub-Sub communication solutions cannot be used for utility M2M applications that require a broad spectrum of QoS requirements such as delay. Note that some clean-slate research such as Publish-Subscribe Internet Routing Paradigm PSIRP [8] is focused on designing physical switches/nodes that inherently provides the Pub-Sub capabilities. However, this approach is far from practical due to interoperability issues.

Before proceeding with the detailed discussion, the context for the physical connectivity between M2M devices/machines and SDN switches must be understood. One class of machines such as RTUs, PMUs, or utility-side application servers are directly connected to SDN enabled switches. However, the other class of machines such as meters, DER sensors, and EV chargers are indirectly connected to SDN switches due to the small volume of data traffic generated and received by these machines. Typically, these machines communicate with SDN switches with low-bandwidth and high-delay access technologies such PLCs (Power Line Communications) or unlicensed radios. Thus, for the class of machines typically found in a distribution network of the power grid, wired/wireless access-points (i.e., customer edge nodes) are used to directly connect them with an SDN switch. However as the data carried on these networks are expected to be part of the critical data for maintenance of the grid (as in the case of demand response), it is important that end-to-end data security should be ensured irrespective of physical connectivity. As a result, access points as middle-boxes are not allowed to manipulate data from end M2M devices.

B. Overview of SVUN

Fig. 4 represents the overall idea of the SVUN that consists of M2M clients (as publishers or subscribers), M2M control nodes, and a set of SDN switches (*i.e.,* OpenFlow switches [3]).

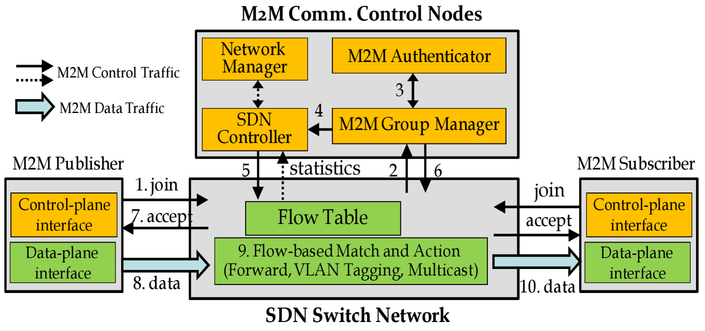


Figure 4 Schema of an Instance of Our SVUN.

SVUN data plane has the following three key notions: (1) Layer-4 flow match for switch access ports, (2) VLAN identifier tagging/stripping [1] for switch trunk ports, and (3) Pub-Sub communication groups.

SVUN control plane consisting of M2M group manager, M2M authenticator, SDN controller [9], and so on provides in a dynamic and fine-grained manner membership management and authentication measures for establishing secure and QoS-aware utility M2M communications (*i.e.,* VLAN per pub-sub group), compared to alternatives such as IEEE 802.1X or VMPS [10] that provides membership management and authentication measure in a static and coarse-grained manner. Please refer to our prior work [11] for the details of M2M control plane.

We emphasize that our unique contribution against other SDN work achieves the complete automation of connectivity/security configuration by combining M2M group mangers with a SDN controller. Compared to other Pub-Sub work, our approach has the following distinct features: 1) line-speed packet processing and forwarding, 2) per-group VLAN traffic isolation, 3) per-group QoS management (*i.e.,* delay-sensitive), and 4) traffic-flow monitoring for load balancing and fail-over.

C. Details of SVUN

SVUN by design addresses the scalability issue as messages of multiple VLANs can be concurrently traversed over switch access ports as well as switch trunk ports. Moreover, we prevent the propagation of security threats that may be developed by an attacker who compromises a member in a VLAN. As shown in Fig. 4, the SDN enabled switch network can perform line-speed message processing and forwarding because of Ternary Content Addressable Memory (TCAM) lookup speed. As default, a Pub-Sub group is represented as a VLAN graph spun across the utility's communication network. A vertex in the VLAN graph is an SDN switch that has a VLAN flow entry in its flow table[2].

Path Establishment

The SVUN has two distinct path establishment processes.

(1) The path establishment for M2M control flows: we note that communications (*i.e.,* control traffic such as join messages) among M2M control nodes (specifically between M2M group managers and SDN controllers) requires paths across a set of SDN enabled switches. In the event of encountering a new M2M control flow, the M2M control path can be reactively established as OpenFlow SDN switches can inherently handle unmatched

[2] In physical SDN switches, a flow table is typically implemented in TCAM and Static Random Access Memory–SRAM. The former is for checking packet match and the latter for performing actions.

messages. Of course, it is possible to have proactive path set up for the M2M control flow due to the small size of M2M control nodes (*i.e.,* in the order of tens). One advantage of the proactive control paths is that the communication delay for the services can be significantly reduced since we can avoid the non-negligible delay[3] of the reactive path setup.

(2) The path establishment for M2M data flows: establishing communication paths for M2M data flows is always triggered in a reactive manner by M2M control messages such as join messages from M2M clients (as publishers or subscribers) over a control path that has already been established. As described at the beginning of this section, M2M application groups are being configured per pub-sub group, and so a data plane path is formed as a unidirectional network graph that consists of SDN enabled switches and communication links. For an application group *T* and its network graph *G*, graph *G* is updated (*i.e.,* addition, modification of flow entries) whenever a M2M client joins or leaves group *T*.

For the establishment of data plane flow paths, a “packet-in” message [2] is never triggered in our SDN switches. This results in a significant reduction of the SDN control traffic load caused by the SDN flow establishment. This is an important distinction against the conventional SDN notion.

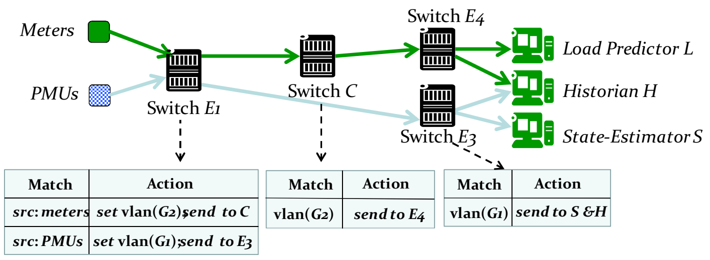
Scalability

The SVUN is highly scalable with respect to flow entries used for SDN switching. The memory (*i.e.,* TCAM and SRAM) of SDN switches where flow entries are kept is a major network resource as most OpenFlow SDN switches available in today’s market have small-size flow tables (*i.e.,* less than 4K flow entries). The scale of flows for M2M control traffic in a SVUN is independent of the number of application groups and group participants. It only depends on the number of M2M control nodes required to set up the service and the number of SDN switches that are associated with M2M group managers. A small number (*e.g.,* in the order of tens) of M2M control nodes is sufficient to support utility M2M communications.

The scale of flows for M2M data traffic in an SVUN is dependent on the number *K* of application groups and the number *M* of group participants. However, the scaling impact of the number of group participants is bounded by the number of SDN switches *N* due to the effect of VLAN aggregation and multicast in SDN switches (see Fig. 5 and 6). Since an application group is represented as a VLAN, the SDN switch has only one VLAN flow entry in its flow table. That is, the maximum number of flow entries per SDN switch is O(*K*) and *K* is independent of *N* and also typically smaller than *M.* This scaling is a unique characteristic of the SVUN.

For a utility and its application group, the SVUN can optimize the number of hops (or end-to-end delay) for the application group using network resources already allocated to the utility, since the application group’s VLAN graph is recomputed whenever a participant joins or leaves the group. As a

[3]It is typically less than 100ms; however, more than 1 second under heavy reactive setup requests, according to our measurement performed over physical OpenFlow SDN switches [3]. We notice that processing OpenFlow “packet-in” messages creates a non-negligible work load in SDN switches and would be a significant factor in rule setup delay.

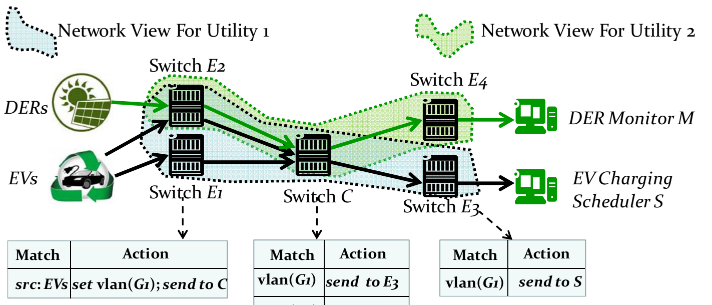


Match	Action
src: meters	*set* vlan(*G2*)*; send to C*
src: PMUs	*set* vlan(*G1*)*; send to E3*

Match	Action
vlan(*G2*)	*send to E4*

Match	Action
vlan(*G1*)	*send to S &H*

Figure 5 Intra-virtualization Scenario, a utility operates different applications such as a PMU group *G*1 and a metering group *G*2.



Match	Action
src: EVs	*set* vlan(*G1*)*; send to C*

Match	Action
vlan(*G1*)	*send to E3*
vlan(*G2*)	*send to E4*

Match	Action
vlan(*G1*)	*send to S*

Figure 6 Third Party -Virtualization Scenario, one utility with EV charging group *G*1 and the other utility with DER group *G*2.

consequence, the increase of end-to-end delay with growth in the number of group participants can be avoided.

Automatic Network Virtualization

We demonstrate utility network virtualization scenarios using four example utility M2M applications: metering-based demand prediction; PMU measurement based real-time state estimation; DERs real-time monitoring; Plug-in EV charging control. Fig. 5 shows a virtualization scenario within a utility. Fig. 6 shows a network virtualization scenario across multiple utilities.

The virtualization within a utility is available by M2M group management alone (see Figure 4). As illustrated in Figure 5, a utility operates PMU measurement data based real-time state estimation group *G*1 and metering data based demand prediction group *G*2 over a subset of SDN enabled switches that have been given to the utility. In the presence of failure of SDN enabled switches or utility servers, we can perform fail-over using the remaining switches or backup utility servers. The number of flow entries that each SDN switch holds is bounded by the number of VLANs (i.e., the number of application groups), as already described. In group *G*1, there are a number of PMU devices and two utility-side servers: 1) computation server *S* that performs grid state estimation in real-time and 2) storage server *H* that persistently keeps all data generated by the utility. Group *G*1 is represented as a VLAN (*G*1) graph across the given subset of SDN enabled switches. As shown, switches *E*3 and *E*4 have to support VLAN multicast feature; otherwise, the number of flow entries in the switches must increase to two. In group *G*2, there are a massive number of AMI meters and two utility-side servers: 1) computation server *L* that performs demand load prediction and 2) storage server *H*. Group *G*2 is represented as VLAN (*G*2) graph across the given subset of SDN enabled switches. The major difference between these two application groups is end-to-end delay handling. In VLAN (*G*1), the stringent delay requirement of real-time state estimations must

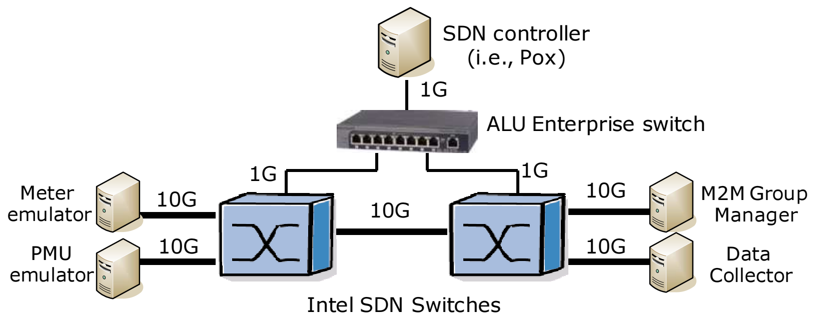


Figure 7 Our Lab test-bed with two SDN switches.

be met. In VLAN (*G*2), the delays are not critical but the graph size needs to be minimized due to the massive number of AMI meters that can unnecessarily consume expensive memory resources on the given subset of SDN enabled switches.

Virtualization across multiple utilities is also possible by appropriate network resource and group management (see Figure 6). As shown in Figure 6, two utilities can virtually operate an EV charging scheduler group *G*1 and DER monitoring group *G*2 respectively over their respective subset of SDN enabled switches (four for utility 1 and three for utility 2). The number of flow entries that each edge SDN switch holds is bounded by the total number of VLANs across utilities. We remark that SVUN allocates one VLAN per application. Some VLANs for applications such as AMI metering are under single utility and by contrast other VLANs for applications such as synchophasor measurements, plug-in EV charging are established across multiple utilities. Thus, the number of flow entries that each edge SDN switch holds is dependent of the number of applications rather than the number of utilities, In fact, a portion of the flow table per SDN switch can be statically or dynamically sliced for each utility. Compared with *G*2, *G*1 needs mobility support since a plug-in EV can communicate with different edge SDN switches over time due to its nomadic plug-in behavior. The SDN flow management for mobile application groups must be efficient on aspects of memory utilization without violating delay requirements. Fortunately, since today's EV battery charging speed is slow, the end-to-end delay requirement for plug-in EV charging application is not too stringent. This implies that communication paths for EV charging can be established reactively since edge SDN switches can inherently handle unmatched messages.

V. Feasibility Study

In this section, we discuss a real implementation of SVUN written in Python and C++ and tested in our Lab test bed.

A. Implementation

In our prior work [11] without using the SDN concept, we have already implemented all M2M control components (except a SDN controller) using C++. In the prior work, for a pub-sub group, a M2M group manager (as a message broker) is chosen by using an overlay technology [12], and forwards both M2M control traffic and M2M data traffic. However, the prior work is limited to meet stringent delay requirements since the message broker's data forwarding engine was built as a user-mode

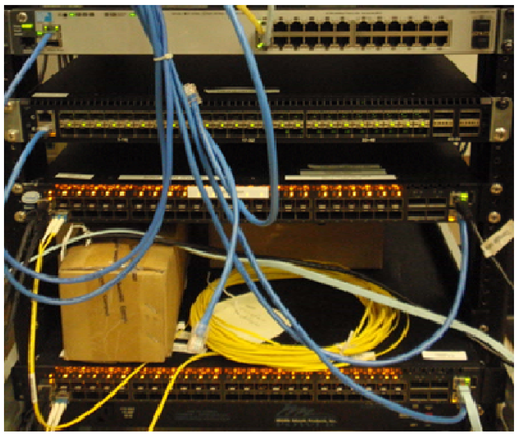


Figure 8 A picture of SDN switches in our Lab test-bed.

software module. In this work, we have thus upgraded the prior implementation to significantly improve the delay performance of M2M data traffic. In this implementation of SUVN which the SDN concept is applied to, a M2M group manager does not handle M2M data traffic anymore; it requests a SDN controller [9] of installing VLAN flow-rules and also let M2M publishers know where publishing data is about to be sent; as a result, for a M2M data traffic over a pub-sub group, it is forwarded by a set of SDN switches in a data-path established for the group.

B. SUVN Lab Test-bed

For our SUVN Lab test-bed, two Intel SDN switches [3] (supporting OpenFlow ver. 1.0.0) and five PCs have been used, as shown in Fig 7. A PC plays as a M2M group manager, a SDN controller, a 100-meters emulator (as publishers), a 100-PMUs emulator (as publishers), or a data collector (as subscribers). As we focus on data-plane communication delay other than control-plane communication delay, the M2M authenticator is omitted.

We have two testing scenarios: 1) a meter joins a meter data collection group in a serial fashion (every two second interval), 2) a pair of a meter and a PMU joins a meter data collection group and a PMU data collector in a serial fashion respectively. Under the testing scenarios, we measured three metrics: 1) flow-table occupancy per switch, 2) end-to-end delay of M2M data plane and 3) M2M control plane. The flow-table occupancy shows scalability of our SVUN since the TCAM is critical but has limited resources; the end-to-end delay of M2M data plane is the time difference between when an M2M publisher sends data and when an M2M subscriber receives the data. This metric corresponds to the forwarding performance of SDN switches in M2M data traffic. The end-to-end delay of M2M control plane is the time difference between when a M2M client (as either a M2M publisher or a M2M subscriber) sends a join message to its M2M group manager and when it receive an accept message from its M2M group manager.

C. Measurement Results

Flow Table Occupancy: the maximum number of hard-state flow rules for M2M data traffic is only two irrespective of the number of M2M devices. There exist a small number of soft-state flow rules (deleted after a timer is expired) for connectivity between devices and an M2M group manager.

Delay on aspects of M2M data plane: In principle, once VLAN flow rules for M2M data-plane have been installed, we see line-speed packet lookup and forwarding of TCAM. In the implementation of SVUN, we observed that the end-to-end delay from publishers to subscribers is never more than 150 ns irrespective of the size of data.

Delay on aspects of M2M control plane: M2M control traffic delay is either about 30ms or about 90ms. Compared with M2M data traffic delay, it is fairly high, even though it is tolerable. We observed the following delay sources: 1) flooding-based ARP discovery, 2) TCP connection setup between M2M clients and M2M group manager, 3) A VLAN flow setup for data-plane.

VI. Conclusion

In this paper, we show that conventional virtual networking technologies including IEEE 802.1Q is limited to support large-scale utility M2M applications such as demand response based on AMI metering, real-time estate estimation based in PMU measurements, or grid-operation aware EV charging. Motivated by the limitation, we propose new SDN-based architectural solution for virtual utility networks (SVUN) that will support self-configurable, secure, and scalable deployment of utility M2M applications. Through measurements performed over a small-but-real Lab test-bed, we confirm the feasibility of the SVUN. Further, we note that the proposed SVUN with small modifications can be extended to a more general solution that supports other M2M applications beyond Smart Grid context.

References

[1] IEEE Std. 802.1Q-2011, Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks.

[2] OpenFlow Switch Specification Version 1.0.0, Dec. 2013.

[3] R. Ozdag, Intel® Ethernet Switch FM6000 Series, Intel white paper, 2012.http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/ethernet-switch-fm6000-sdn-paper.pdf.

[4] Budka, K., Deshpande J., and Thottan, M, Communication Networks for Smart Grids – Making Smart Grid Real, Springer, 2014.

[5] Y.-J. Kim, V. Kolesnikov, and M. Thottan, TSAF: Tamper-resistant and scalable mutual authentication framework for plug-in EV charging, IEEE SmartGridComm, Oct. 2013.

[6] B. Aboba, D. Simon, and P. Eronen, Extensible Authentication Protocol (EAP) Key Management Framework, IETF 5247, Aug. 2008.

[7] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec, The Many Faces of Publish/Subscribe, ACM Computing Surveys, vol. 35, no. 2, June 2003.

[8] P. Jokela, A. Zahemszky, C. E. Rothenberg, S.a Arianfar, P. Nikander, LIPSIN: line speed publish/subscribe inter-networking, ACM SIGCOMM Conference on Data communication, Aug. 2009.

[9] POX Controller, https://openflow.stanford.edu/display/ONL/POX+Wiki.

[10] OpenVLAN Member Policy Server, http://sourceforge.net/projects/vmps.

[11] Y-J. Kim, J. Lee, G. Atkinson, H. Kim and M. Thottan, SeDAX: A secure, resilient and scalable platform, IEEE JSAC, vol. 30, no. 6, July 2012.

[12] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A geographic hash table for data-centric storage," in Proc. ACM WSNA Workshop, Sep. 2002.

[13] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, Fabric: A Retrospective on Evolving SDN, ACM HotSDN Workshop, Aug. 2012.

[14] E. Rosen, A. Viswanathan, and R. Calton, Multiprotocol Label Switching Architecture, RFC 3031, IETF 2001.

APPENDIX: An SDN Solution with MPLS

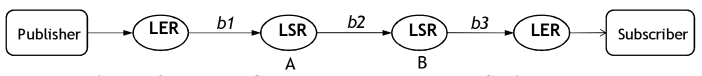


Figure 9 LER, LSR, Labels, and Label Switched Path.

We have so far considered virtual utility networking on SDN edge switches that are connected to M2M clients. In this section, we discuss how our SDN solution can be integrated with non-edge switches that provide connectivity among edge switches.

Advantages of using Multi-Protocol Label Service (MPLS)-based SDN design are discussed in [13]. MPLS provides natural tools for virtual network slicing – which is necessary for implementing a network architecture for M2M communication. The SDN edges switches can be implemented using Label-Edge Routers (LER), whereas the non-edge switches in the network are implemented using Label-Switched Routers (LSR) Of MPLS Architecture [14]. So, the host-network interface is completely independent of the packet–network interface between a packet and the LSR [13]. The SDN controller will need to configure the required Label switched paths between the LERs going through the core network made entirely of the LSRs or the non-edge switches.

The edge switches (LERs) are responsible for mapping the destination addresses into the "labels" configured by the SDN controller. The non-edge switches (LSRs) only need to look at the labels and forward the packet to the next switch on the LSP, swapping out the label as configured by the SDN controller while configuring the LSP. Consider an LSP configured from an LER to another LER, as shown in Fig. 9. The labels are configured by the SDN controller in the switches. The ingress LER, adds Label *b*1 to the incoming packet from, say, a publisher connected to it. LSR A after receiving the packet looks into its label table and decided to forward the packet to LSR B with label *b*1 swapped with label *b*2. And so on. Finally, the egress LER removes the label and forwards the packet to the subscriber.

Note that in an actual MPLS network, the LSPs are created by the LERs and LSRs using one of the several MPLS control plane protocols such as Label Distribution Protocol (LDP) or Resource Reservation – Traffic Engineering (RSVP-TE). In the SDN environment, LSPs should be configured by the SDN controller as needed.

As discussed in [13], the main advantage of using MPLS-like SDN deployment is that, the core network of LSRs is very simple and does not even have to know the protocols used by the hosts between them or between a host and an edge switch. On the other hand, the LERs can be complex and need not know the actual networking implementation of the core as long as destination addresses and mapped to the labels. In principle, the LERs and the core networks can be independently designed, implemented, and/or upgraded.

The virtual network slices implements by SDN will be equivalent to well understood L1, L2, and L3 MPLS services (also known as MPLS VPNs) emulating the respective protocols. All advantages of traffic isolation, per service QoS, and reliability properties of the MPLS services are instantly available to the SDN network