

Next Generation Emergency Communication Systems via Software Defined Networks

Milos Manic¹, Dumidu Wijayasekara¹, Kasun Amarasinghe¹, Joel Hewlett¹, Kevin Handy¹,
Christopher Becker¹, Bruce Patterson², Robert Peterson³

¹Department of Computer Science,
University of Idaho
Idaho Falls, Idaho, USA
misko@ieee.org

²City of Ammon,
Technology Dept.
Ammon, Idaho, USA
bpatterson@ci.ammon.id.us

³ Special Projects,
Albion Telephone Company
Albion, Idaho, USA
ccp@jdcinc.net

Abstract— The existing Emergency Communication System (ECS) infrastructure is becoming increasingly outdated with many members of the public moving away from landline based telecommunications and broadcast television in favor of cellular telephones and internet-based streaming entertainment services. Current systems for public services such as E911 and Emergency Alert System broadcasts are no longer a reliable means for reaching the public. In addition, both wired and wireless telecommunications systems can become overwhelmed, as was the case following Hurricane Katrina in 2005 and the World Trade Center disaster in 2001, and in fact, when communications are needed most urgently, the difficulty of maintaining effective communication increases exponentially. While the use of Internet based alternatives could resolve some of these problems, existing Internet infrastructure offers no dedicated or priority bandwidth to the user for emergency communications (e.g. E911 or Emergency Alert System). The current Internet capacity can also be overloaded due to high volume network data streams. Under these conditions, emergency communications (e.g. inbound and outbound communications reporting catastrophic or emergency events) may have their packets dropped resulting in incomplete and/or delayed communications. To alleviate these problems, this paper presents a novel framework for ECS using network virtualization via Software Defined Networks (SDN). A table top demonstration of ECS using SDN was developed at the University of Idaho, Idaho Falls. This paper details the foundational technologies and overviews the steps taken at the University of Idaho to develop ECS using SDN.

Keywords—Emergency communication systems, software defined networking, network slicing, OpenFlow, cloud computing.

I. INTRODUCTION

Reliable and fast Emergency Communication Systems (ECS) are an important factor in timely and effective disaster response [1], [2]. Fast response is critical in disaster situations as a significant portion of fatalities occur in the first few hours of the disaster and the first 72 hours after the disaster has been identified as the most critical period for surviving victims [1], [3].

Due to the multitude of disasters that claimed many lives in the recent past such as September 11 attacks in the United States of America (2001), Tsunami in Asia-Pacific (2004), Hurricane Katrina in the United States of America (2005), the

Fukushima incident in Japan (2011) and numerous earthquakes around the world, the limitations of existing ECS has been exposed and focus on more reliable ECS infrastructure has been increased [4], [5].

While there are many issues in the existing ECS infrastructure, 3 main problems can be identified [6], [7]. 1) **Lack of capacity**: with a low normal utilization and extremely high peak utilization, it is difficult to allocate bandwidth. Thus, in disaster situations, communication interruptions due to congested networks are often observed. 2) **Incompatible systems**: the systems of different agencies such as law enforcement, fire and health are completely different. This is also true of users who utilize different networks for communication. 3) **User operability**: as systems become more complex and heterogeneous, it becomes difficult for the average user to operate effectively. Out of these problems, the problem of capacity is especially difficult to handle as majority of time (i.e. non-disaster times) the network utilization is extremely low. However, in a disaster situation the network usage is extremely high. Thus, allocating bandwidth required for effective ECS is problematic [7], [8].

In addition to these problems, the existing ECS in the USA is suitable for landline and broadcast media based communication. Thus, the current trend of people moving towards cellular communication in favor of landlines and streaming entertainment in favor of broadcast media [9] makes it difficult to reach the public in a disaster situation.

Internet based solutions can alleviate some of these problems. However, existing Internet infrastructure does not provide dedicated bandwidth or priority based communications to users such as homes, businesses and public buildings. Furthermore, the current Internet capacity can also be overloaded due to large downloads; file sharing (e.g. Dropbox, Microsoft OneDrive), backups (e.g. Mozy, Carbonite), entertainment streaming (e.g. Netflix, Hulu), online gaming (e.g. Xbox-live, PlayStation Network), etc. Thus, Internet based ECS can experience loss of Quality of Service (QoS) or service disruptions due to high volume internet traffic.

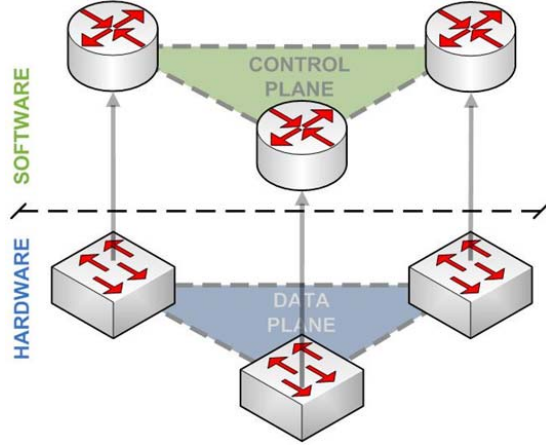


Fig. 1 Decoupling of the control plane and the data plane via SDN

Therefore, this paper presents a novel framework for ECS using network virtualization via Software Defined Networking (SDN) [10]. The presented framework utilizes OpenFlow [10] to decouple the control plane and the physical infrastructure to achieve virtualization of the network. This enables control changes to be made at the software level, alleviating the need for hardware changes to update network configuration. Thus, the presented ECS using SDN framework enables dynamic bandwidth allocation as well as priority based packet forwarding. Isolation of services is achieved by means of Virtual LANs (VLANs). Isolation of service and dynamic bandwidth allocation ensures QoS even at disaster times. Furthermore, virtualization of the network via SDN makes the system scalable as new services can be added on-the-fly

The rest of the paper is organized as follows. Section II introduces network virtualization via Software Defined Networking. Section III briefly discusses the evolution of the current Emergency Communication System. Section IV presents the architecture of Emergency Communication Systems via Software Defined Networking. Finally, Section V provides specific details about implementation and, Section VI concludes the paper.

II. INTRODUCTION TO NETWORK VIRTUALIZATION VIA SOFTWARE DEFINED NETWORKING

Traditional network architectures are becoming increasingly inadequate in meeting modern day networking requirements [10]. Prevalent architectures are primarily hardware centric and as a result, their configurations are vendor specific [10], [11]. Thus, it is almost impossible to control and update it centrally. As a result, no abstractions exist and whenever a networking related problems arise, it is resolved by protocols which are built from scratch. Therefore, the application programmers should be aware of and should adjust to, the infrastructure of the network and its vendor specific configurations.

Software Defined Networking (SDN) is a Networking paradigm which decouples the control pane from the data plane of the network [10] (see Fig. 1). In other words, SDN provides a layer on top of the network infrastructure layer,

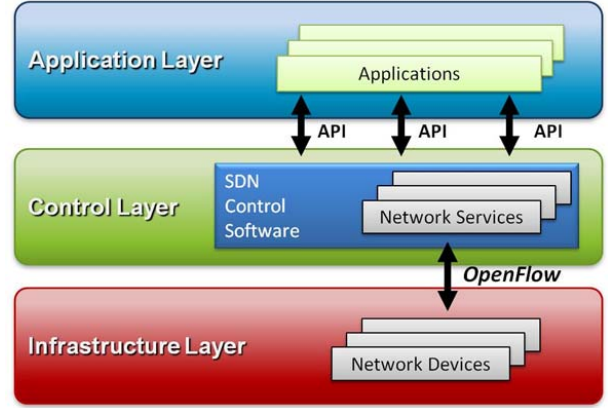


Fig. 2 Abstraction of SDN architecture

which provides control instructions to the infrastructure (see Fig. 2). Thus, the switches in the network become simple packet forwarding devices. As a result, the control of the network become logically centralized [10], [11] and the need to customize control rules for vendor specific configurations are alleviated. Further, this results in providing the network applications an abstraction over multi vendor network infrastructure, which results in easier development and evolution.

Once the data plane and the control plane is decoupled into two layers, the communication between the two layers are handled through a **Control Data Plane Interface(CDPI)** [10]. The first standard CDPI was OpenFlow and is currently used as well [10], [12]-[14]. OpenFlow controller provides control instructions to OpenFlow enabled network hardware.

SDN and OpenFlow make the networks programmable and provide centralized control of multi vendor environments. Since the networks become programmable, the network can be virtualized for better control. In this paper, the following consequents of network virtualization which is introduced by SDN and OpenFlow are largely used.

A. Isolation of Service

Software Defined Networking allows the isolation of services or classes of services by making services or classes of services reside in their own virtual channels. To the service provider, this appears as a private network which is shared only with end users who are using that particular application. Since the service providers do not have to accommodate traffic from other applications, the flow and protocol of the dedicated channel can be updated and optimized to the application itself, without interfering other services.

B. Controlled Quality of Service & Optimization of Traffic Flow

The isolation of services also allows for control of the network Quality of Service (QoS). By providing high bandwidth applications with their own dedicated channel, traffic on the virtualized network is restricted to the associated application.

Each application or class of applications is provided its own channel. Therefore, the traffic flow on the virtual network can be optimized based on the needs of the application itself. In OpenFlow, this can be done via the “Flowvisor” [15], [16]. In the Flowvisor approach, the translation unit acts as a protocol proxy and is shared between multiple switches and multiple controllers [16].

C. Scalability

SDN separates the control layer from the infrastructure layer. Therefore, infrastructure of the network can be scaled up without affecting the existing system.

Furthermore, since the services or applications are provided their own channels and they are controlled using OpenFlow independently, new services and end users can be added on the fly to the system without affecting the existing system. As an example, new servers and switches can be introduced to the system to accommodate new services and end users without affecting the existing system.

III. EVOLUTION OF EMERGENCY COMMUNICATION SYSTEMS

This section details the evolution of ECS and the drawbacks associated with the prevalent ECS.

The introduction of the dial tone and direct dial service in the 1950s, paved the way for the user to directly connect to the emergency responder or an operator to request assistance. To further improve the system, in 1968, a three digit emergency number; 9-1-1 (911), which worked well with the phone systems which in place at that time, was introduced. At this point, 911 was a basic service that needed no extra processing, but with time, as the usage of the system expanded, the vulnerabilities of the system were exposed.

In 1996, several incidents which showed the inadequacy of the prevalent system forced the Federal Communications Commission (FCC) to announce a mandate for an enhanced 911 service [17]. This resulted in the Wireless Communications and Public Safety Act of 1999 which mandated 911 services to be included for non landline phones as well. This service was named Enhanced 9-1-1 (E911) [18].

With the introduction of E911, wireless devices had the ability of placing emergency calls. Therefore, it included source location determination of emergency calls. In order to keep track of locations, each signal carrier was responsible for keeping an Automatic Location Information (ALI) database. Different type of carriers had to use different techniques to maintain the ALI database which lead to complexities in the system. For instance Voice over Internet Protocol (VoIP) providers use a packet switched network as a gateway to the Public Telephone Network. Here, the VoIP provider keeps track of the subscriber's connected device, not the location.

Further, with the rapid adoption of cellular wireless networks, location determination and information dissemination to mobile devices have become complex. Currently, ECS are moving towards Short Message Service (SMS), Reverse 911 etc. to disseminate information.

But it is evident that ECS has moved from simple and basic services to very complex services which rely on multiple databases, a multitude of stakeholders and instantaneous data processing and routing to function properly. The multiple points of failure that the complexity of these systems has introduced have increased the vulnerability of the system.

IV. GENERAL ARCHITECTURE OF ECS USING SDN FRAMEWORK

This section details the presented framework for ECS using network virtualization via SDN.

The presented framework consists of service providers who provide different services, users who use these services, and the network through which the user and service provider communicate (see Fig. 3). Each service or application is given its own virtual “channel”. This channel encapsulates the entire service from end-to-end in a completely virtualized fashion. This encapsulation isolates the services on the service provider end, through the network, and on the user end device (see Fig. 3). In the presented framework, SDN is used for dynamic, priority based packet switching and virtualization of the network while Virtual LANs (VLANs) are used for isolation of services.

The presented framework proposes a methodology for isolation and virtualization of the service at each of these three levels. The virtualization and isolation enables achieving the required QoS, traffic flow control, and scalability.

A. Network

As mentioned, each service is given its own channel. These channels enable isolation of data flow. In the presented framework this is achieved by using Virtual LANs (VLANs) for each service [19].

The virtualization of the network is achieved via SDN. The data flow in the presented framework uses OpenFlow to determine packet forwarding rules, which separates the hardware and the control plane of the network. Separation of the control plane and hardware results in adding a virtualization layer to the network. Using OpenFlow, it is possible to dynamically update packet forwarding rules. Thus, packet priority as well as bandwidth can be changed according to current requirements.

B. User End

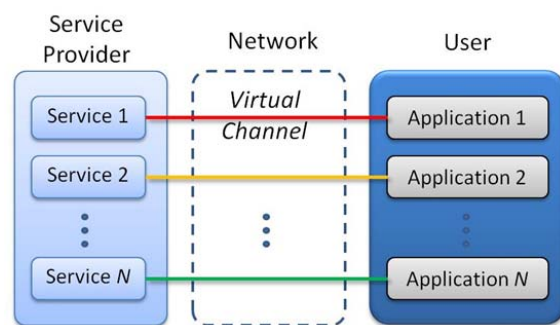


Fig. 3 ECS using SDN basic framework

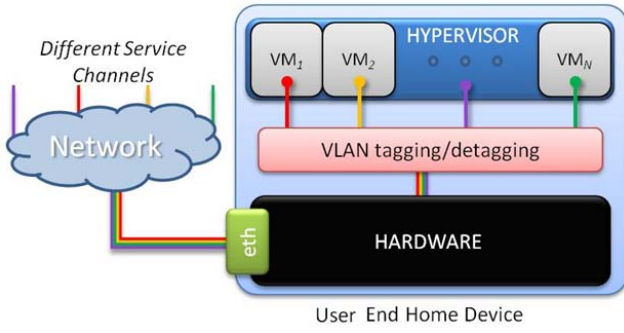


Fig. 4 User end of the presented framework

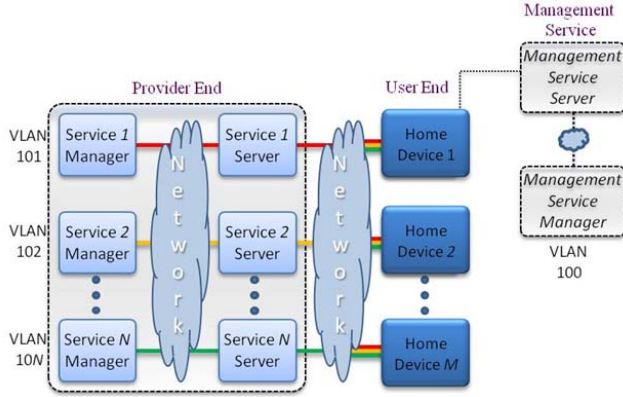


Fig. 5 Overall architecture of the presented framework for ECS using SDN

The User End of the presented framework is depicted in Fig. 4. Virtualization on the user end device is accomplished through the use of Virtual Machines (VMs). The use of VMs that can each run different operating systems will allow each application developer to choose whichever environment is most appropriate, familiar, or conducive to their own application. Also, because VMs share the available resources through the use of a hypervisor that virtualizes the hardware, hardware emulation can be used to make the applications highly flexible with respect to the architecture of the host system.

Isolation on the user end device will also be achieved through the use of VMs. Each application will be presented on its own VM. This will keep system or software issues from one service, from impacting the operation of the other services. If an application crashes, or freezes a system, the effects will be isolated to the associated VM, which may be shut down or restarted without impacting the VMs running the other applications.

C. Provider End

At the provider end, virtualization and isolation of the services is handled in the same way as the end user, using VMs. Each application is hosted on its own server and is implemented as a VM. This allows multiple application servers to be hosted on one or more physical servers. This implementation allows for load balancing by dividing the user load among the application servers. If demand increases, new servers can be easily added. In addition, if a server goes down

due to malfunction or maintenance requirements, the load can be seamlessly offloaded to the other available servers. This also allows for isolation of services. If a system error occurs on one of the virtual servers, it can be shutdown or restarted without disturbing the operation of another.

D. Overall Architecture

The proposed overall architecture is shown in Fig. 5 and contains three primary components; 1) User End, 2) Provider End, and 3) Management Service. The User End is composed of *Home Devices* and, as the name suggests, they are a devices that can be placed at a client's home or workplace, and is responsible for displaying the messages or content sent from a particular service or application. A Home Device contains a VM for each service and a Hypervisor which is responsible for controlling VMs such as suspending all the VMs except for the one linked with the service that is active. Even though a VM for each application resides on the Home Device, the content that is displayed resides on the provider end. Thus, the applications are virtualized.

The Provider End is comprised of service providers and each service provider is composed of 1) *Service Manager* and 2) *Service Server*. The Service Manager is responsible for determining the content of the service that is displayed on the Home Devices. The Service Server is responsible for communicating with the home device to send the aforementioned content. In the proposed architecture, the Service Servers are hosted in the form of VMs and each service resides in its own virtual network (VLAN). Hence, communications of each service is also isolated. When the content is updated from the Service Manager, it communicates with the Service Server. Then, The Service Server communicates with the Home Device to update the content at the user end.

The Management Service is a special purpose service. The primary objective of the Management Service is to dictate the service or application that is active at a particular point in time. Since it too is a service, it resides in its virtual network and consists of its own manager and server. The manager of the Management Service selects the service that needs to be displayed on the Home Devices. Then, the server of the Manager Service communicates with the Hypervisor which resides in the home devices and specifies which VM should be kept active. This enables emergency services to be activated and displayed on the User End when needed.

E. Advantages of the Proposed Architecture

The communication between Service Servers and Home Devices is performed according to rules dictated through OpenFlow. Furthermore, VMs that exist on the end user, enables each application or service to communicate with and send content to a separate VM. Thus, the presented architecture performs virtualization of the network from the service provider all the way through to the application, past the user's network hardware. This enables each service or application to reside in its own channel which leads to isolation of services.

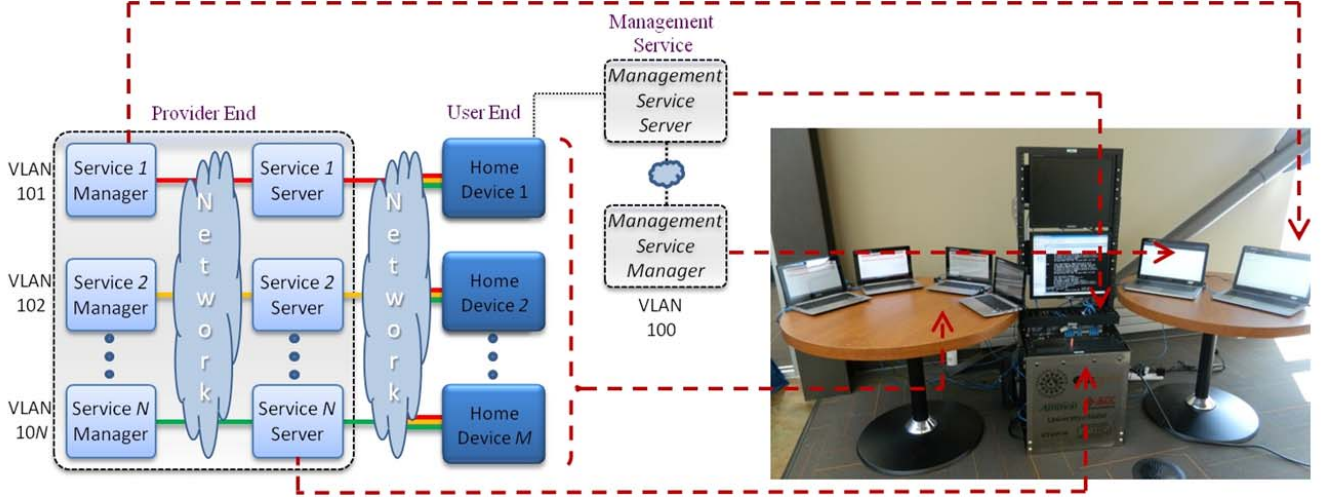


Fig. 6 Implementation of the presented framework for ECS using SDN at University of Idaho, Idaho Falls

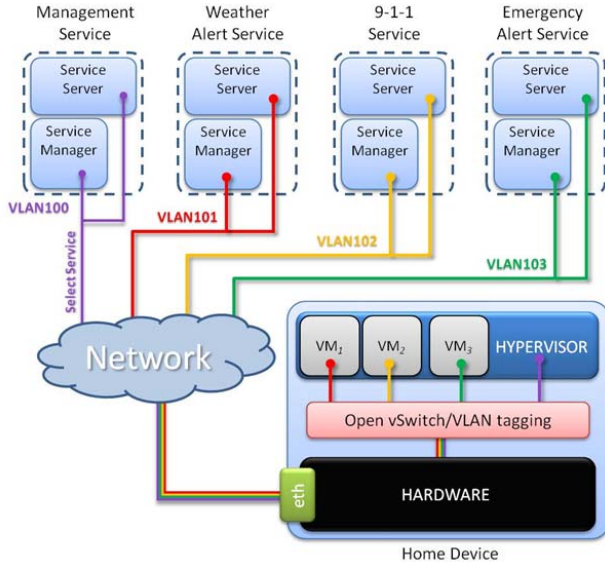


Fig. 7 Implementation of the network architecture of the presented framework

Each application or service, communicating end-to-end via its own channel ensures Quality of Service (QoS) for the application. Further, it uncovers the possibility of controlling parameters for each application or service independently without affecting other services such as controlling the bandwidth dedicated to each application and packet forwarding priorities.

Since independent channels exist between each service and the client, new services and clients can be added to the system on the fly making the system scalable. Since OpenFlow is used for packet routing, the core of the network need not be changed when services and users are added to the system.

V. ECS USING SDN IMPLEMENTATION

The architecture stipulated in Section IV was implemented as an experimental test bed by the University of Idaho, Idaho

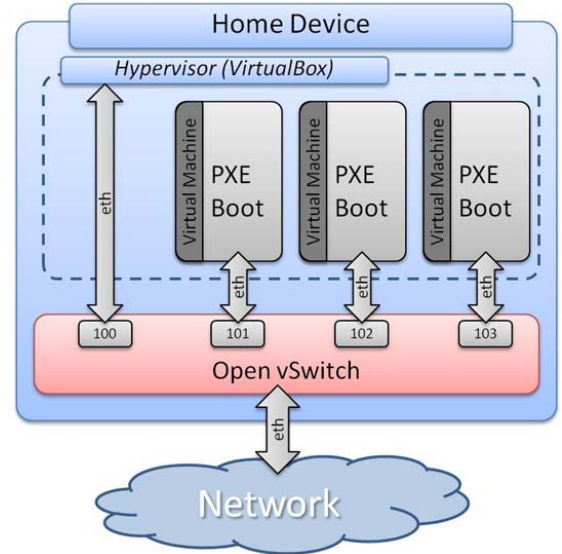


Fig. 8 Implementation of the user end of the presented framework

Falls. Fig. 6 shows the experimental test bed and its' components as related to the presented framework.

Four services were implemented to test the proposed architecture; 1) Management Service, 2) Weather Alert Service (WAS) 3) Alternative 911 Service (A911) Emergency Calling, and 4) Emergency Alert Service (EAS). Management Service was responsible for controlling the Hypervisor in the client machine. WAS and EAS were designed to send warnings, alerts and news to the users. The 911 service was designed so that the client/user could contact the emergency responder via a direct channel.

A. Network Implementation

The network packet forwarding was carried out using a HP E3800 OpenFlow switch. Floodlight OpenFlow controller [20] was used to control the OpenFlow switch. Each of the four services was designed to reside in their own virtual network (VLAN) and the Service Server and Service Manager of each

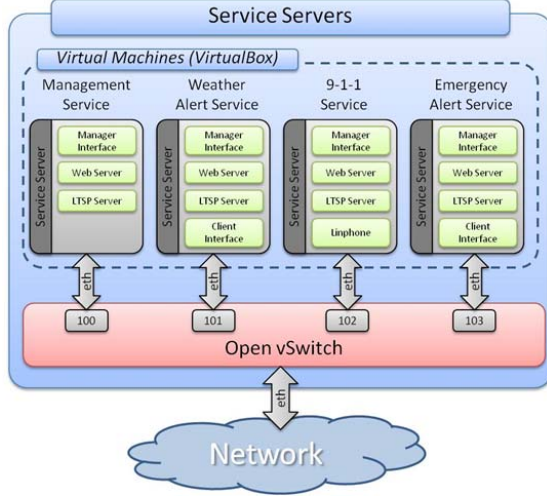


Fig. 9 Implementation of the provider end of the presented framework

service was contained in the associated VLAN. VLAN tagging [19] was used to identify the traffic between different services. The flows were specified on the controller depending on the VLAN tags of different service. For instance, Weather Alert Service was assigned VLAN tag 101 and Alternative 911 was assigned VLAN tag 102. Fig. 7 depicts the data flow implementation in detail.

B. User End Implementation

In the experimental test bed, the user end was implemented as four Home Devices. A VM for each service was placed on the client machines and a Hypervisor was used to switch between the VMs when the Management Service dictated a change. Four laptop computers were used as client systems which had similar processing power and performance capabilities. VirtualBox, which is a virtualization product [21], was used to contain the VMs on the clients and VirtualBox acted as the Hypervisor as well. When the Manager of the Management Service dictates a change, a PXE (Pre-boot Execution Environment) boot was utilized to activate the VM of the associated service.

OpenvSwitch, which is an OpenFlow enabled virtual switch, was used to perform packet forwarding inside the Home Device to send it to the appropriate VM. This again, was controlled using OpenFlow rules associated with the VLAN tag of the data packet. For instance, a data packet arriving at the Home Device with the VLAN tag 101, is sent to the VM associated with WAS.

C. Provider End Implementation

The Provider End was implemented for the aforementioned services using four VMs. The VMs were designed to act as the Service Servers. For the implementation purposes, the VMs were hosted on a single system. The Service Manager systems of services WAS and EAS, were designed to communicate with the Service Server using a web interface. For service A911, VOIP client which is under the GNU Public License; Linphone, was used.

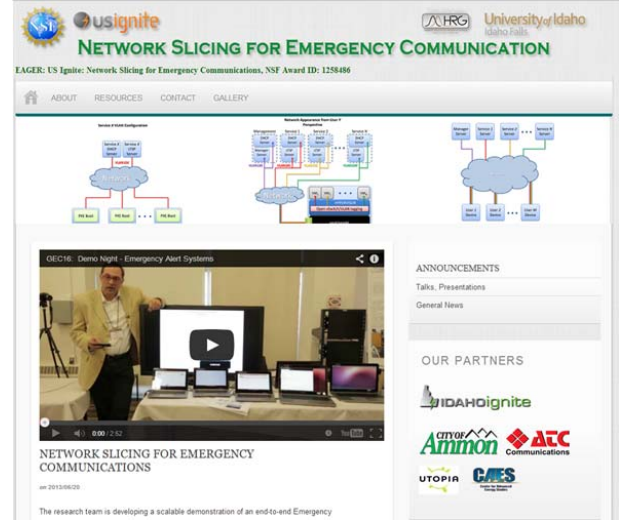


Fig. 10 The dedicated website providing implementation and configuration details of the presented framework (<http://nsec.if.uidaho.edu/>)

The Service Server was designed to contain the interface shown on the Home Device as well the interface shown on the Service Manager. A LTSP server was included in the Service Server to enable the VM on the Home Device to view content which is on the Service Server.

Since all Service Servers were hosted in the same system in the form of VMs, OpenvSwitch was used for the same purpose as it was used in the Home Device implementation.

D. Documentation

Implementation details and all necessary documentation are provided in the dedicated website [22]. A screen capture of the dedicated website is shown in Fig. 10. The website provides configuration documents and specific details about the implementation and required hardware so that the presented framework can be implemented elsewhere. Furthermore, a Sourceforge page is also set up so that the software can be downloaded and updated [23].

VI. CONCLUSION

This paper presented a novel framework for Emergency Communication Systems (ECS) using network virtualization via Software Defined Networks (SDN). The presented method utilizes SDN to implement dynamic bandwidth allocation and priority based packet switching for resilient ECS. Furthermore, Virtual LANs (VLANs) are used to isolate services. The paper details the framework and specific implementation details of the table-top demonstration developed at the University of Idaho, Idaho Falls.

As future work, the presented framework will be migrated to GENI for large scale implementation and testing. Performance analysis will be done to evaluate the usability of the presented framework in a real-world scenario. Furthermore, the possibility of utilizing the presented framework for other commercial uses that require dynamic bandwidth allocation and high quality of service will be explored.



Fig. 11 The interface for the Manager of the Management Service



Fig. 12 The interface of the Service Manager for Weather Alert Service

APPENDIX

A. Software

Below are the software programs which were utilized in the implementation of the SDN based ECS test bed given in this paper.

- *Floodlight Controller:*

Floodlight controller is a Open SDN controller which is under the Apache-License. Floodlight controller was used to provide the open flow rules for packet forwarding.

- *VirtualBox:*

Virtual Box is a virtualization software under the GNU General Public License. It was used to contain and manage the VMs used in the implementation. Further, it was used to act as the Hypervisor on the client side.

- *LTSP - Linux Terminal Server Project*

LTSP is a software which is distributed under the GNU General Public License which adds thin client support to LINUX servers. In the implementation, an LTSP server as used in each Service Server to provide the content to the Home Device VMs.

- *LinPhone:*

LinPhone is a VoIP application which is distributed under the GNU General Public License. Linphone was used as the calling interface in implementation of the Service A911.

B. Hardware

Below are the special purpose Hardware Used for the implementation of the aforementioned implementation.

- *HP E3800 OpenFlow switch*

HP 3800 is a 24 port OpenFlow switch and it was used for packet forwarding using OpenFlow. The OpenFlow controller provided the packet forwarding rules.



Fig. 13 The interface for the Home Device when a Weather Warning is issued using Weather Alert Service

C. Screen captures of implemented systems.

Fig 11 depicts the interface designed for the Manger of the Management Service. By pressing the button shown on the interface, different service can be made active at the user end.

Fig 12 depicts the interface of the Service Manager for the WAS. Using the interface shown the service provider has the ability of updating the content that is being displayed at the Home Device when WAS is active.

Fig 13 illustrates the interface of the Home Device, when WAS active and a weather warning is issued.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (NSF), under EAGER: US Ignite: Network Slicing for Emergency Communications, NSF Award ID: 1258486.

REFERENCES

- [1] Z. Shao, Y. Liu, Y. Wu, L. Shen, "A Rapid and Reliable Disaster Emergency Mobile Communication System via Aerial Ad Hoc BS Networks," in *Proc. of Int. Conf. on Wireless Communications, Networking and Mobile Computing*, pp. 1-4, Sept. 2011.
- [2] M. Dervin, I. Buret, C. Loisel, "Easy-to-Deploy Emergency Communication System Based on a Transparent Telecommunication Satellite," in *Proc. of Int. Conf. on Advances in Satellite and Space Communications*, pp. 168-173, July 2009.

- [3] Y.-N. Lien, L.-C. Chi, Y.-S. Shaw, "A Walkie-Talkie-Like Emergency Communication System for Catastrophic Natural Disasters," in *Proc. of Int. Conf. on Pervasive Systems, Algorithms, and Networks*, pp. 309-314, Dec. 2009.
- [4] T. Li, M. Huang, J. Chang, J. Shi, J. Yang, J. Yu, J. Hu, "A pilot emergency communication system based on IP and airship," in *Proc. of Int. Symp. on Antennas Propagation and EM Theory*, pp. 1008-1011, Dec. 2010.
- [5] H. Okada, H. Oka, K. Mase, "Network construction management for emergency communication system SKYMESH in large scale disaster," in *Proc. of IEEE Globecom Workshops*, pp. 875-880, Dec. 2012.
- [6] P. Conder, L. Linton, M. Faulkner, "Cognitive radio developments for emergency communication systems," in *Proc. of IEEE Symp. on New Frontiers in Dynamic Spectrum Access Networks*, pp. 658-659, May 2011.
- [7] P. Conder, "Improving interoperability on a National level for Law Enforcement, Public Safety and Security communications," Report for the Law Enforcement & Security Radio Spectrum Committee, Victoria University, Sept. 2008.
- [8] Y.-N. Lien, L.-C. Chi, C.-C. Huang, "A Multi-hop Walkie-Talkie-Like Emergency Communication System for Catastrophic Natural Disasters," in *Proc. of Int. Conf. on Parallel Processing Workshops*, pp. 527-532, Sept. 2010.
- [9] M. Brown, "Abandoning the news," in *Carnegie Reporter*, vol. 3, no. 2, pp. 2-11, 2005.
- [10] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks", ONF White Paper, Apr. 2012.
- [11] H. Kim, N. Feamster, "Improving network management with software defined networking," in *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114-119, 2013.
- [12] E. Haleplidis, S. Denazis, O. Koufopavlou, J. Halpern, J. H. Salim, "Software-Defined Networking: Experimenting with the Control to Forwarding Plane Interface," in *Proc. of European Workshop Software Defined Networking*, pp. 91-96, Oct. 2012.
- [13] J. Crowcroft, H. Oliver, Y. Bar-Geva, "Tearing down the Protocol Wall with Software Defined Networking," in *Proc. of IEEE SDN for Future Networks and Services*, pp. 1-9, Nov. 2013.
- [14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, "OpenFlow: enabling innovation in campus networks," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, 2008.
- [15] P. Skoldstrom, K. Yedavalli, "Network virtualization and resource allocation in OpenFlow-based wide area networks," in *Proc. of IEEE ICC Workshop on Software Defined Networks*, June 2012.
- [16] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar. "Flowvisor: A network virtualization layer," OpenFlow Switch Consortium, Tech. Rep, 2009.
- [17] Reed, Jeffrey H., Kevin J. Krizman, Brian D. Woerner, and Theodore S. Rappaport. "An overview of the challenges and progress in meeting the E-911 requirement for location service." *Communications Magazine*, IEEE 36, no. 4 (1998): 30-37.
- [18] Public Law, "Wireless Communications and Public Safety Act of 1999," Communications, 1999.
- [19] S. Viswanath, P. Chow, "Multiport data switch having data frame VLAN tagging and VLAN stripping," U.S. Patent No. 6,151,322. 21 Nov. 2000.
- [20] Project Floodlight, Software Defined Networks [Online] Available: <http://www.projectfloodlight.org/floodlight/>
- [21] Virtualbox, [Online] Available: <https://www.virtualbox.org/>
- [22] Network slicing for Emergency Communication [Online] Available: <http://nsec.if.uidaho.edu/>
- [23] Sourceforge, Network slicing for Emergency Communication [Online] Available: <http://sourceforge.net/projects/nsec/>