REF: Enabling Rapid Experimentation of Contextual Network Traffic Management using Software Defined Networking

Lyndon Fawcett, Lancaster University, UK
Mu Mu, The University of Northampton, UK
Bruno Hareng, Hewlett Packard Enterprise, France
Nicholas Race, Lancaster University, UK

Abstract

Online video streaming is becoming a key consumer of future networks, generating high-throughput and highly dynamic traffic from large numbers of heterogeneous user devices. This places significant pressure on the underlying networks and can lead to a deterioration in performance, efficiency and fairness. To address this issue, future networks must incorporate contextual network designs that recognise application and user-level requirements. However, designs of new network traffic management components such as resource provisioning models are often tested within simulation environments which lack subtleties in how network equipment behaves in practice. This paper contributes the design and operational guidelines for a Software-Defined Networking (SDN) experimentation framework (REF), which enables rapid evaluation of contextual networking designs using real network infrastructures. Two use case studies of a Quality of Experience (QoE)-aware resource allocation model, and a network-aware dynamic ACL demonstrate the effectiveness of REF in facilitating the design and validation of SDN-assisted networking.

Introduction

With the growing popularity of video services and the increasing online presence of traditional broadcasters, online video is believed to be the leading consumer of future networks, generating high-throughput and highly dynamic network traffic [1]. Adaptive media such as HTTP adaptive streaming (HAS) using protocols like TCP or Quick UDP Internet Connections (QUIC) is becoming the de-facto standard for online media streaming. The non-cooperative and unsupervised resource competition between adaptive media applications leads to significant detrimental quality fluctuations and an unbalanced share of network resources [2]. Therefore, it is essential for content networks to better understand the application and user-level requirements of different data flows and to manage the traffic intelligently. Traditional network traffic management approaches based on the configuration of proprietary devices are cumbersome and inefficient in the dynamic management of network resources [3]. Software defined networking (SDN) is a network paradigm that decouples network control from the underlying packet forwarding. It continues to gain traction as a vehicle for delivering efficient and flexible context-aware network programming. OpenFlow, first introduced by McKeown et al. [4], is commonly used to realise the concepts of SDN, with many networking devices now supporting the protocol. For every rule match specified, OpenFlow automatically maintains and updates packet counters, which may be interrogated on demand by an OpenFlow application. Furthermore, with the introduction of Fog

Computing and Network Function Virtualization (NFV), the cloud is being brought closer to the user in the form of micro data centres or cloudlets [5]. This opens compute locations that are close to the edge, such as Customer Premises Equipment (CPE), to enable contextual network traffic management services that process and can enforce at the network edge [6]. Context-aware networks are different from traditional networks as they are aware of the flows that have passed through the network, and can make decisions to alter the network based on this information.

Distinctions with network emulators and SDN facilities

Recently there has been pioneering work on exploiting SDN for traffic engineering and network management. Nam et al. [7] propose an SDN application to monitor streaming flows in real time, dynamically changing the routing paths for better user experiences. Akella et al. [8] harness SDN to provide QoS bandwidth guarantees for priority users through a mathematical model. Mehdi et al. [9] argue for using SDN as a security mechanism for the home through anomaly detection and remediation. Wong et al. [10] proposes to solve peak-hour broadband network congestion problems by pushing congestion management to the network edge using a two-level resource allocation design. However, SDN-assisted novel network programming models are often designed and tested in a simulation or emulation environment such as Mininet [11]. Whilst these test environments do offer a means of experimentation, they do not consider the effects that network protocols, client programs, hardware limitations, physical switches and other real-world factors may have on the outcomes. Major design flaws may be masked during simulation or emulation and are only discovered in prototyping or early production phases. Emulations can also be limited by the capabilities of software switches such as Open vSwitch.

Many researchers and projects have recognised the following benefits of an experimental testbed that provides an environment close to that of production networks: 1) proving that SDN applications will operate with real-world hardware, or testing the behaviour of specific hardware in each experimental context; 2) experimenting with specific operating system stacks, and their network implementations; or 3) supporting experiments where hardware constraints (CPU, memory, etc.) are part of the variables under evaluation. Facilities such as Fed4FIRE [12] and their tools provide a means for many researchers to run SDN experiments over geographically distributed hardware which would otherwise not be possible. However, when slicing the networking resources between multiple users, the outcomes can change on each experimental run due to the load generated by simultaneous experiments, ultimately skewing results. Further impacting this, each experimenter is unaware of the other ongoing experiments, meaning that it is difficult to determine if the results you received were as expected or due to another user on the facility conducting a load intensive experiment.

The contribution in this article differs from existing facilities and software in various ways, one area where REF excels is in its flexible and portable deployment method; a network tested on the experiment facility can also be tested within Mininet, or even executed in production with little changes. As well as this, contrasting to existing facilities that typically provide very detailed low-level control to just the network infrastructure, REF provides higher level abstractions of both the network and virtualisation infrastructures through orchestration, automating the creation, connection, running, and cleaning of nodes in an experiment. Furthermore, it also provides abstraction over the network for making the creation of context aware traffic management applications as streamline as possible. Additionally, with the unique configuration using slicing and port multiplexing, REF can create much larger physical networks with limited hardware than

its competitors. Finally, the entire REF framework can be used and modified by anyone without any kind of registration or subscription to a federation.

In this article, *REF is introduced*, an experimentation framework and a guide to building a testbed that together provides a blueprint for an SDN-based contextual network design facility. Firstly, it describes the framework, covering the requirements of the framework then the purpose of each component within the system as well as the abstractions that it provides to the user. Next the experiment testbed is detailed, providing a guide on how to construct your own virtualisation and network infrastructure for experimentation. After this, both use cases are described and used to show REF in operation, this includes a Quality of Experience (QoE)-aware resource allocation model, and a network-aware dynamic ACL. Finally, the article goes into a discussion on interesting findings that arose during the creation and use of the system.

REF Experimentation framework

Setting up a functional SDN testbed is a challenging process requiring extensive knowledge and experience. We aim at creating a framework that assists the researcher in creating their own SDN applications and experiments, whilst providing isolation to avoid conflict between experiments. Furthermore, the framework should make the most of the hardware available, so that researchers can create topologies to a similar scale that are available in simulation environments. Additionally, the framework should allow replicating large-scale localised experiments, and this is useful when modelling a datacenter, home, or business topology where there is a dense collection of nodes with low latency between each other. This feature is generally not available using a shared testbed due to the equipment being geographically distributed.

To provide a harness capable of supporting rapid deployment and orchestration of experiments, an experimentation platform will need to fulfil the following requirements:

- *Experiments close to practice and at scale.* The system should be able to realise and manage a large number of clients and networks. Meanwhile, to provide both realism and scale, the environment will encompass both physical and virtual elements.
- *Dynamic manipulation of the network*. Rate limiting, queuing, flow redirection, and other features of SDN implementation are required to enforce decisions made by intelligent network traffic management modules.
- *Configurable clients.* The client's configuration (image and resources) should be quickly changeable (automated based on test manifests) after an experiment to set up for a new experiment as well as at run time.
- *Rapid repeatability of experiments in a clean environment*. Ensure that no residual effects are left over from previous experiments by removing VMs and networks before a new experiment.

Functional components

The REF framework (Figure 1) orchestrates the virtual and physical network infrastructure to assist the execution and statistical data gathering of network based experiments. It consists of a three-layer architecture: the top layer contains components provided by the researcher including the test manifest and application/user-level functions such as our case studies: QoE and security applications. The middle layer contains the REF orchestrator which interfaces with, and includes,

the infrastructure managers. The bottom layer contains the network and virtualisation infrastructure where the experiments are deployed.

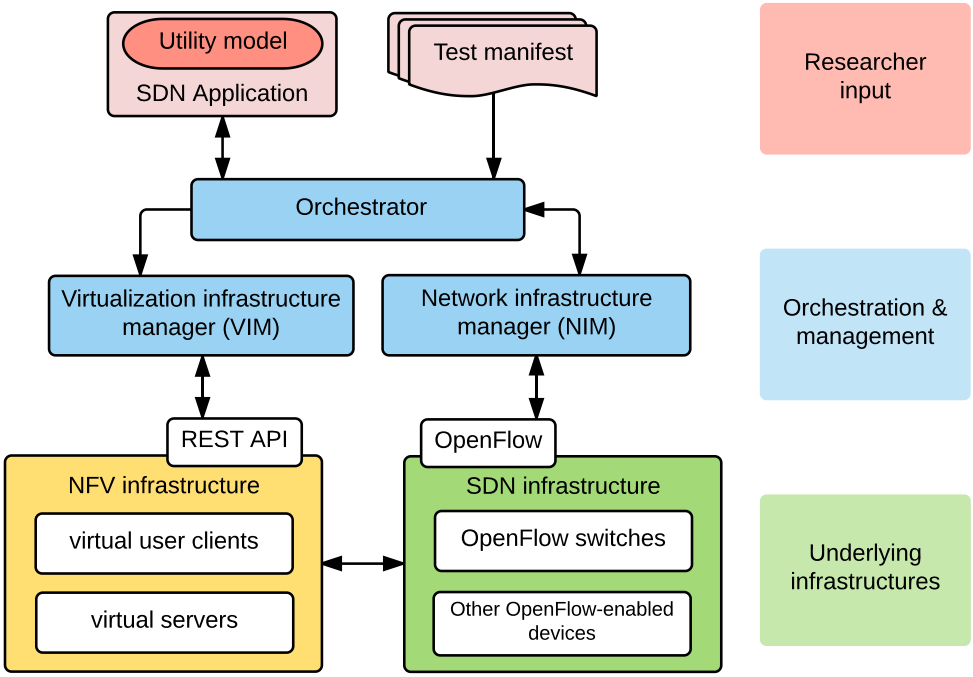

Figure 1. Rapid experimentation framework

The test manifest describes the experiment in a JSON format. It includes each of the client's IP address, the networks each is attached to, the virtual machine image to be used, and network emulation requirements. The example manifest below shows two networks *lan1* and *lan2* who share the same aggregation network (*group1*) and the currently available bandwidth on the aggregation network is configured to be less than the sum of bandwidth on *lan1* and *lan2*.

```
Spec = {
        'name' : "test experiment"
        'keypair' : "openstack_rsa"
        'controller' : "10.30.65.210"
        'credentials' : {'user' : "Test", 'password' : "Test", 'project' : "Test"},
        'networks' : [{'name' : "lan1", "subnet" : "192.168.1.0/24", "rate" : 5000, "group" : 1},
                {'name' : "lan2", "subnet" : "192.168.2.0/24", "rate" : 5000, "group" : 1}]
        'groups' :    [{'id': 1, rate: "8000"}]
        'hosts' : [{'name' : "h1", 'image' : "Scoot", 'flavour': "small", 'net' : [{lan1}]},
                {'name' : "h2", 'image' : "Scoot", 'flavour': "small", 'net' : [{lan2}]}]
}
```

The SDN application contains a utility model which captures application-level requirements such as QoE and security measures. As part of the framework, the SDN application is an interchangeable component which communicates with the REF orchestrator through a Remote Procedure Call (RPC) interface providing information about resource allocation on flows. Additionally, information is sent back in regards to the current throughput at different points in the network using SDN-specific control messages such as OpenFlow's *meter statistics* and *flow statistics* messages.

The REF Orchestrator handles communication between all the components. It includes two subcomponents, the Virtual Infrastructure Manager (VIM) and Network Infrastructure Manager (NIM). VIM controls the virtualization infrastructure through a RESTful API, it launches and configures experiment nodes with information from the test manifest. At the end of the experiment it resets the test environment by triggering VIM and NIM clean methods, removing networks and virtual machines it instantiated, so that the environment is ready for the next experiment.

NIM controls the network infrastructure and consists of a Ryu OpenFlow controller containing a metering and monitoring application. It installs meter flow mods on request from the SDN application and provides information from the network including current throughput of flows and switches. These abstractions over the network infrastructure are interfaced directly with the orchestration component which in turn provides a simple RPC API to the researcher's SDN application. This allows the orchestrator to define and configure network setup on-the-fly through a simple JSON formatted request. A typical request would be to report the current network traffic level for a port or previously defined flow. An example command would be to define a flow (e.g. source/destination IP pair), and request that the flow is limited to a certain level (defined in Mbps). The response to this command includes a unique identifier which can be used in subsequent requests for traffic data. VIM is positioned above the virtualisation infrastructure (managed by OpenStack), and provides an interface to the orchestrator to provide an instantiation of experiment nodes that are connected to the experiment network. The network infrastructure creates connections between nodes and switches and provides a platform for configuring link bandwidth.

REF Abstractions

The design for the REF architecture was an iterative process based on initial requirements for context-aware SDN network applications. These desired requirements included: port and flow monitoring, total bandwidth capacity estimation, and controlling bandwidth on a per flow and port basis. We then added functionality to support other state-of-the-art applications created by other researchers using the framework; this included a collection of metering statistics, enabling the ability to define the flow granularity instead of using the same as the forwarding application, and the ability to provide and choose from a catalogue of existing forwarding applications. The design of REF stemmed from our experience working with other testbeds and frameworks including Fed4FIRE.

The following lists the main abstractions provided by REF that SDN applications can use. These features are available through a JSON-RPC interface between the researchers SDN application and REF's orchestrator.

Virtualisation and node management abstractions:

- Creating and destroying VMs after each experiment and during when required.
- Executing scripts on each client for the experiment.
- Recording and aggregating experiment logs from nodes.
- Configuring link bandwidth between virtual nodes.

Network traffic management abstractions:

- The monitoring of flows at multiple levels while simultaneously logging these for post-experiment analysis.

- The monitoring and logging of throughput observed on switch ports.
- Providing network forwarding by default, thus reducing the time and difficulty to researchers when creating their utility application.
- Enforcing throughput constraints on flows, groups of flows, ports, and groups of ports.
- Monitoring and logging of OpenFlow meter counters.
- Configuring link bandwidth between physical nodes.

The feature list of REF is continuously evolving as SDN matures, for an extensive and current list of the capabilities of this framework, consult the public project webpage for REF (http://lyndon160.github.io/REF/).

Building an experimentation testbed

To demonstrate the effectiveness of REF, we provide an implementation guideline (Figure 2) and two experimental case studies based around the delivery of video to multiple home environments and another based around a smart grid network. For both cases, these connections share a restricted link to the Internet. In addition, the CPEs and the local DSLAM are also under SDN control to provide programmable link configuration for dynamic management of traffic. This reflects our vision of an SDN-assisted pervasive computing and networking environment, allowing granular network control at the very edge of the network.

Virtualization Infrastructure

At the core of the virtualisation infrastructure is an OpenStack installation. This provides the means of building and connecting virtual machines (VM) to instantiate a significant amount of dynamically configurable live client applications. The OpenStack installation is standard, with one main modification: VLAN trunks are used to break-out network interfaces from virtual machines. These are then mapped one-to-one to exclusive physical interfaces on a switch. We refer to this process as *port-multiplexing*, as it allows an Ethernet switch to implement remote physical interfaces for virtualized machines. This is an essential feature for our experimentation as it allows each client to be directly assigned to a physical port on an SDN controlled switch. The mechanism for this is based on the use of VLANs to carry VM traffic onto the switch. The configuration is such that each VM is allocated an exclusive OpenStack (Neutron) network.

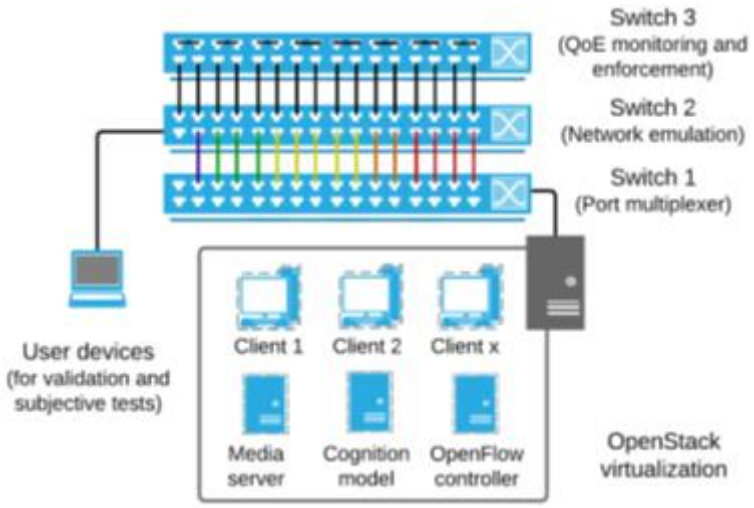

Figure 2. Virtualization and network infrastructure

The setup and management of this infrastructure are controlled by VIM. As such, we use an in-house orchestration tool titled MiniStack (https://github.com/hdb3/ministack). Its purpose is to bring the network and client creation automation capabilities shown in Mininet. MiniStack provides the ability to rapidly build, reconfigure, and delete (all within seconds) experimental topologies using a simple and extensible configuration format which include networks, connections and clients. Furthermore, as this is a modular component, it can be used by other projects to automate creation and deletion of network topologies.

Network Infrastructure

The network infrastructure used in this example consists of two OpenFlow v1.3 capable switches (Switch 2 and 3 shown in Figure 2) with metering support. In our facility, we use Hewlett Packard Enterprise Aruba's 3800 series (HPE3800) switches, as they fulfil both requirements. However, other compliant switches could be used instead, including OpenFlow switches from Pica and Corsa. The HPE3800 also hosts other important capabilities, such as the ability to flexibly partition a single physical switch into several virtual OpenFlow switches. Each of these is a complete and distinct OpenFlow instance. This is outside of the scope of OpenFlow, but is a feature present on many devices available on the market. This partitioning feature is vital in achieving the scale required in experimentation without incurring the associated cost. However, it is important to note that the switches memory is shared between instances, reducing the maximum number of flow entries per application. For flow table efficiency, the roles of switch 2 and 3 can be merged by using a switch with multi-table support.

Use case study 1: QoE-aware resource allocation

We use the evaluation of UFair [13], a QoE model, as a use case study of how REF supports rapid research and experimentation. UFair seeks to reduce the frequency of adaptations over a group of HAS clients, and moderate individual clients' choice of stream bandwidth, to the benefit of all applications on the same network. The core of UFair is a mathematical specification for the optimal bandwidth to be consumed for each member of a group of clients, based on the prevailing network resources and user device capabilities. It is stateful, to retain data about past forced bandwidth changes and thus reduce the impact of resolution changes across the entire client group. UFair operates by using REF's monitoring and enforcing capabilities to get information about the network status and "capping'' resources on individual media streams, with the assumption that media clients can adapt their bandwidth utilisation in response to network constraints. Therefore, resource allocation or other traffic control can be achieved transparently in the network without cooperation from user applications. The effectiveness of such network-based control is dependent on how application and user-level context is incorporated in network traffic management design and executed by SDN.

Experiment topology and operation

Figure 3 depicts a tiered topology representing a multi-household network. Each of the households contains 4-6 hosts, all of which are connected to a gateway. This gateway is then connected, along with other gateways in the topology, to an aggregation switch (switch B). Over

another hop (towards switch A), a foreground and a background server act as endpoints as sources or sinks of respective traffic types.

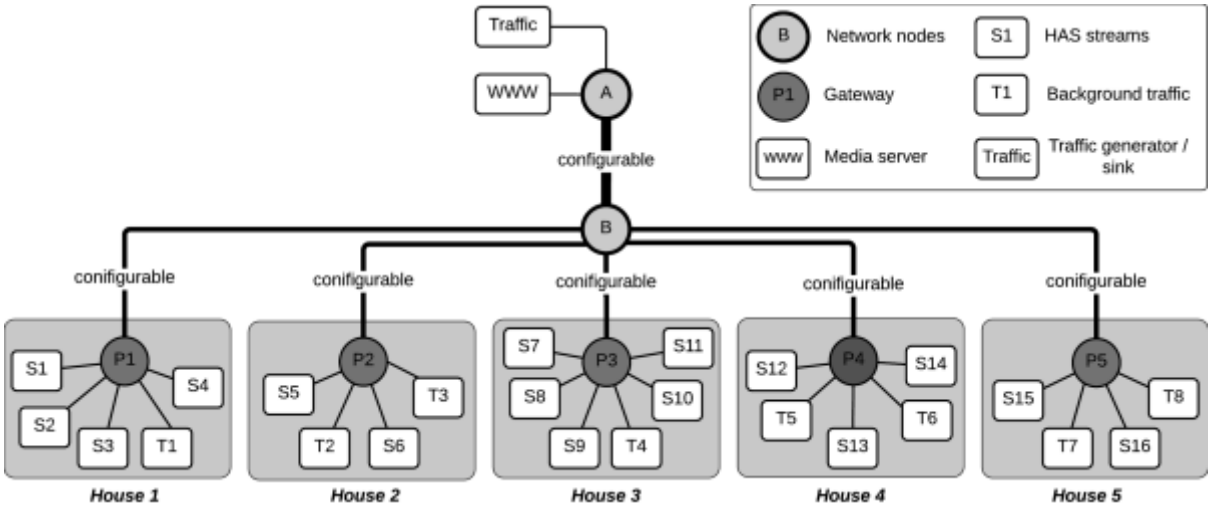

Figure 3. Experimentation topology

To emulate a potential home network environment where a household has limited bandwidth, and the link shared between houses is also limited, network link characteristics are dynamically configured. Through REF, the links between switch B and the gateway switches are limited to 20Mbps. Similarly, the link between B and A is restricted to 50Mbps. The sum of the connectivity available to household links is 100Mbps, and is greater than the link between B and A. This results in a situation whereby there is more demand than there is supply in the case of multiple households. In these circumstances, the adaptive streams in each house are affected by hosts within the same house, as well as the behaviour of hosts in other houses. Using REF, network configuration is directly programmable as an integral part of the framework to capture the complex and temporal dynamic characteristics of real-world complex networks.

In this setup, the hosts in households are configured as online video players to request MPEG-DASH adaptive video content from media servers, with one or two hosts in the same household generating background traffic. The experiment used REF to monitor the network statistics of each client, as well as each household. This data is then analysed by the UFair model to determine the most optimal resource allocation strategy. Recommendations given by the UFair model is then applied through REF's traffic enforcement functions, including restrictions per flow and household. We also define *baseline* experiment, where network statistics are still monitored but no additional traffic management is applied.

Results

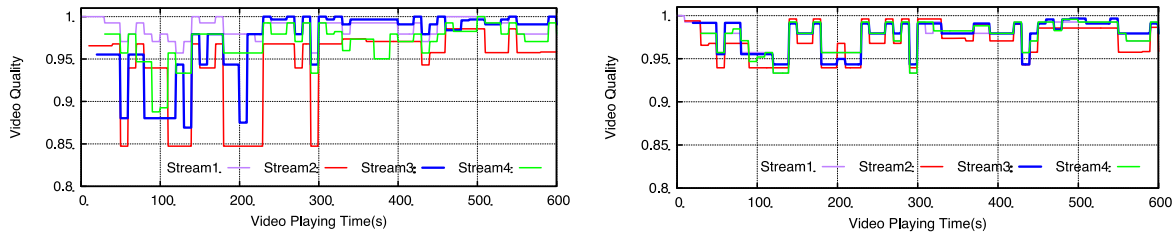

Figure 4. Resource allocation without (left) and with (right) OpenFlow-assisted UFair model

Figure 4 depicts the resultant video quality of each video stream when the OpenFlow-assisted UFair model is inactive and when it is active. Video quality is a rate-distortion function that is used to describe the non-linear relationship between quality and bit-rate [13]. The results clearly demonstrate the significant differences of the network provisioning strategy adopted by the user-level model compared with the conventional TCP-based network-level baseline model. The baseline model allows video streams with more intensive requests at the transport layer to acquire more resources, leading to some video streams being heavily penalized. Using the bespoke UFair model the network management element in the testbed can schedule the resource according to the QoE requirements and link status of every HAS stream. Thus, network resources are dynamically provisioned in a way that similar video quality is maintained on all related media streams for the entire course of the experiment (Figure 4). Furthermore, the UFair model was able to avoid any severe video quality fluctuation due to its awareness of all competing media flows in the same network. In this case, we can validate the performance of a utility model by repeating the test 100 times without human intervention. The functions offered by REF, including streamlining the orchestration of utility model, virtualisation infrastructure, and physical OpenFlow equipment, allows researchers to focus on application-level design.

Use case study 2: Context-aware access control (Smart-ACL)

The Smart-ACL use case study considers how REF supports experimentation with security-based context aware SDN applications. Smart-ACL is designed to provide protection in the network on top of SDN switches, and reflects an increasing research interest in the adoption of SDN within critical infrastructures. This specific use case considers the use of SDN within a Smart Electricity Substation environment, where protection mechanisms are required against attacks such as Denial of Service (DoS). This level of protection is necessary to prevent an unwanted event, such as a mass power outage [14]. Moreover, standards bodies such as the International Electrotechnical Commission (IEC) have strict security and resiliency requirements in place to prevent this. Before SDN can be safely adopted in these networks, the above issues need to be addressed, whilst adhering to IEC standards.

Smart-ACL harnesses multiple OpenFlow features exposed by REF to prevent a multitude of attacks. It operates by using whitelists, rate-limiting on the packet in flow rule, and making remediation decisions based on network context from the REF. Remediation is applied through REF's rate-limiting and blacklists. It takes information from the network about the whitelisted nodes traffic and classifies this as essential traffic. An average of this traffic is then taken into consideration when rate-limiting non-essential traffic. This value is recalculated periodically with various tolerances to ensure that slow attacks are detected. In this case study, we show how REF has been used to assist in the development of Smart-ACL.

In operation, REF is started and manages the network's connectivity. The Smart-ACL application calls the orchestrator's northbound interface to get information about flows in the network, including flow headers and counters. Using this information, it protects whitelisted services by ensuring that there is enough bandwidth available on the network so that they remain uninterrupted. To do this, Smart-ACL takes information from REF about how much of the total available bandwidth is being used by non-essential traffic (flows not in the whitelist) and rate-limits using REF's enforce feature if the traffic exceeds the total bandwidth minus the whitelists required bandwidth. Furthermore, using meter drop counts from REF, the application detects if a flow is not behaving to the network constraints, if the drop rate exceeds the threshold then the traffic is blocked for a short while to allow other non-essential traffic fair use of the available bandwidth.

Experiment topology and operation

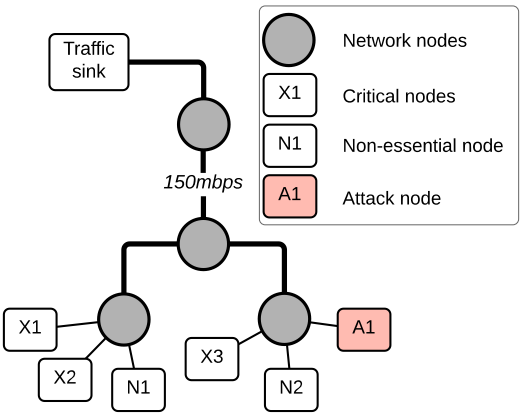

Figure 5. Experimentation topology

Comparable to the previous case study, hosts, attackers, and link limits are applied automatically through a configuration. In this case study, REF was used to automate three essential nodes with a purpose to maintain a connection with the traffic sink generating HTTP traffic at a total target rate of 60Mbps. Alongside these, two non-essential nodes were connected to the traffic sink sending benign traffic. Additionally, there was a single attack node which was generating UDP traffic with no client-side throughput limits across the network towards the traffic sink. Also, a link limit of 150Mbps was set between two of the switches, emulating a constrained environment. Using REF's automation with a test-manifest, this experiment ran for 120 seconds and was repeated 100 times without any additional human intervention.

REF assisted the researchers in developing this application by providing an underlying framework to manipulate the network which also already provided forwarding logic. The use of being able to quickly and automatically repeat the experiment whilst having an output of the traffic in the network assisted when determining thresholds and timeouts for bandwidth rate-limiting flows, allowing the researchers to improve on the application model with ease to ensure that essential traffic remained unaffected by the ongoing attack.

Results

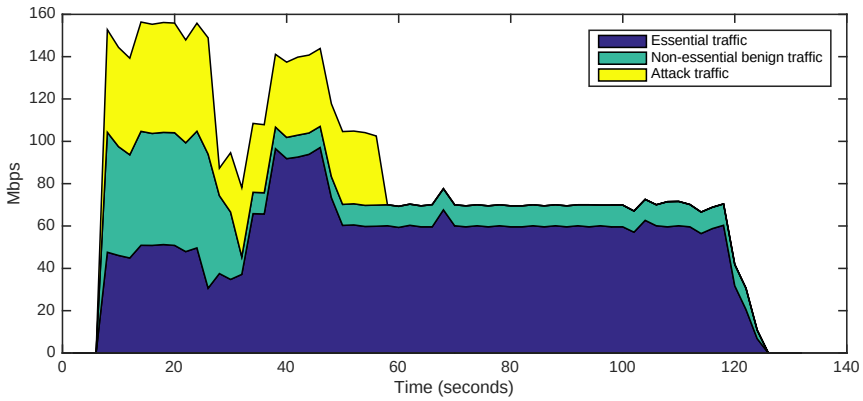

Figure 6. Results with Smart-ACL enabled

Figure 6 depicts a stacked graph of traffic logs produced by REF of the switch located at the top of the topology. The results show the effects that Smart-ACL has on the network when a simple

UDP Flood DoS attack is triggered. These tests were performed 100 times, then each traffic type was averaged. We can see that the essential traffic remains stable, and that attack traffic along with the non-essential traffic was rate-limited from after 25 seconds. The attack traffic was then identified through excessive packet drops on the meter counter and eventually ceased. Without Smart-ACL, the attack would have continued, limiting the bandwidth available to essential traffic. Further information about this case study as well as the code and results are available on GitHub (https://github.com/lyndon160/Smart-ACL).

Discussions

When acquiring OpenFlow-enabled equipment for research and experimentation, it is essential to investigate the advanced features offered by different vendors and on different generations of equipment. The supported OpenFlow version (e.g., 1.0, 1.3, or 1.5) is often a first indication as to the OpenFlow features a device may offer. However, it is unlikely that all *optional features* of an OpenFlow specification will be fully implemented. Furthermore, implementation details of features such as metering are often left open to interpretation for the switch vendors, this can result in experiments behaving differently between two switches with the same advertised capability due to differences in implementations. It is worth investigating differences of device's capabilities and implementation details, as they may have a significant impact on how they support design and evaluation. Thus, we recommend consulting the ONF's OpenFlow conformance list [15] when acquiring a new network switch for an experiment.

Conclusion and Future work

The proliferation of online media is placing tremendous pressure on QoE and security requirements on existing network infrastructure. This has led to a growing body of research developing novel network traffic management models using software defined networking. Many researchers use simulation tools to evaluate their designs, which can overlook effects that are seen in link delay and link bandwidth emulation in networks and clients. This paper introduces REF, a framework that facilitates rapid experimentation of SDN-assisted network designs using a combination of physical equipment and virtualized functions. We carried out two case studies on SDN-assisted QoE and security traffic management applications to validate the REF designs. We also provide detailed guidance and an open-source toolset for the readers to instantiate a research and experimentation environment of their own. By sharing our experiences, we hope to stimulate cross-site interconnected testbeds to support a research and innovation internet environment, enabling new uses of the testbeds and thus research.

Leading on from this research, we intend to continue advancing REF as OpenFlow and its features mature. For vendor-specific features such as packet dropping policies, we are currently in the process of creating drivers for different switches to provide researchers with the ability to easily unlock more of these potentially useful features. Furthermore, we plan to open the network and virtualisation infrastructures more widely as part of a new project, which starts in early 2017. We expect this facility to federate with other testbeds as part of the development activity within the project. Additionally, we are exploring the idea of using creating APIs for other controllers so that a REF application would be portable between controllers. Finally, we will be continually monitoring progress on the state of the art of software switches to one day integrate them with REF for a hybrid infrastructure of virtual and physical switches; this will provide a means to create

experiments at even greater scales without losing experiment rigor. Currently, the REF framework is shared between multiple UK universities in partnership through the UK EPSRC-funded TOCUAN project.

References

[1] Cisco, “Cisco Visual Networking Index: Forecast and Methodology, 2014-2019 White Paper.”

[2] Mu, M., Broadbent, M., Hart, N., Farshad, A., Race, N., Hutchison, D. and Ni, Q., “A Scalable User Fairness Model for Adaptive Video Streaming over Future Networks,” *IEEE J. Sel. Areas Commun.*, 2016.

[3] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, “A Survey on Software-Defined Networking,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[4] N. McKeown *et al.*, “OpenFlow,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 69, 2008.

[5] F. Bonomi, B. Flavio, M. Rodolfo, Z. Jiang, and A. Sateesh, “Fog computing and its role in the internet of things,” in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC ’12*, 2012.

[6] F. M. F. Wong, J.-W. Carlee, H. Sangtae, L. Zhenming, and C. Mung, “Improving user QoE for residential broadband: Adaptive traffic management at the network edge,” in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*, 2015.

[7] H. Nam, N. Hyunwoo, K. Kyung-Hwa, J. Y. Kim, and S. Henning, “Towards QoE-aware video streaming using SDN,” in *2014 IEEE Global Communications Conference*, 2014.

[8] A. V. Akella and X. Kaiqi, “Quality of Service (QoS)-Guaranteed Network Resource Allocation via Software Defined Networking (SDN),” in *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, 2014.

[9] S. A. Mehdi, K. Junaid, and S. A. Khayam, “Revisiting Traffic Anomaly Detection Using Software Defined Networking,” in *Lecture Notes in Computer Science*, 2011, pp. 161–180.

[10] F. M. F. Wong, J.-W. Carlee, H. Sangtae, L. Zhenming, and C. Mung, “Improving user QoE for residential broadband: Adaptive traffic management at the network edge,” in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*, 2015.

[11] B. Lantz, B. Heller, and N. McKeown, “A network in a laptop,” in *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks - Hotnets ’10*, 2010.

[12] “Fed4FIRE Project.” [Online]. Available: http://www.fed4fire.eu.

[13] Mu, M., Broadbent, M., Hart, N., Farshad, A., Race, N., Hutchison, D. and Ni, Q., “A Scalable User Fairness Model for Adaptive Video Streaming over Future Networks,” *IEEE J. Sel. Areas Commun.*, 2016.

[14] A. Johnsson and N. Sigfridsson, “Deployment of smart substation standard IEC 61850,” in *22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013)*, 2013.

[15] “ONF OpenFlow Conformant: Certified Product List - Open Networking Foundation.” [Online]. Available: https://www.opennetworking.org/openflow-conformance-certified-products. [Accessed: 14-Dec-2016].

Biographies

Lyndon Fawcett is a PhD student within the School of Computing and Communications at Lancaster University, he is involved in the EPSRC funded TOUCAN project. His primary research interests are in using Fog computing infrastructures to enhance NFV and SDN. This research encompasses multiple disciplines and entails exploring and creating innovative platforms for network and infrastructure orchestration that maintain an awareness of the underlying network and compute resources available.

Dr Mu Mu is a Senior Lecturer (Associate Professor) at The University of Northampton, United Kingdom. His PhD in Computer Science and Master of Science degree was awarded by Lancaster University, United Kingdom and TU-Darmstadt, Germany respectively. His research interests include software-defined cognitive networking, Quality of Experience, human factors in multimedia systems, and immersive media. Dr Mu has over 50 publications in prestigious conferences and journals. He is a member of IEEE and ACM.

Bruno Hareng is an SDN and Security Solution Manager at HP Networking, Hewlett-Packard Enterprise Aruba EMEA. His graduated from the ENSIMAG Engineering School of Grenoble with an MSc degree in Computer Science and Applied Mathematics. Mr Hareng has more than 20 years of experiences in the industry. He has led many projects on communication networks, software-defined networking, QoS/QoE and security solutions.

Dr Nicholas Race is a Reader in Networking within the School of Computing & Communications at Lancaster University. His research is broadly around experimental networking and networked media, specialising in the use of Software Defined Networking (SDN) and Network Functions Virtualisation (NFV) for new network-level services, including in-network media caching, network-level fairness and network monitoring. His recent work considers the design of NFV service orchestration within Fog Computing environments.