Experiences Monitoring and Managing QoS using SDN on Testbeds Supporting Different Innovation Stages

Stuart E. Middleton, Senior Member, ACM, and Stefano Modafferi
IT Innovation Centre
University of Southampton
Southampton, UK
{sem, sm}@it-innovation.soton.ac.uk

***Abstract*— In recent years there has been a big increase in the number of network-related experiments using software defined networking (SDN) technology. We report on our practical experience over 2 years running network experiments on three classes of testbed facility, each supporting researchers working at a different innovation stage. We run experiments using the commercial Amazon EC2 cloud facility, pre-commercial federated testbed of FIWARE Lab instances and the OFELIA experimental facility. We run an idealized common network experiment on each testbed, reducing its scope where needed to match testbed capabilities, and report details of the practical experience gained using a set of qualitative metrics for direct comparison across classes of testbed. We conclude with a concrete recommendation for pre-commercial testbed facilities to allow better support for network experiments in the future.**

Keywords—Experiment; Software Defined Networking; SDN; Quality of Service; QoS; Testbed; Innovation Stage

I. INTRODUCTION*

Over the last few years there has been a big increase in the number of network-related experiments performed by researchers using software defined networking (SDN) technology. Helping to support this trend a number of testbed facilities are appearing, in both the commercial domain and from national sources such as the European Commission's Future Internet Research and Experimentation (FIRE) programme [7]. In SDN deployments network devices become packet forwarding devices (i.e. the data plane), while the control logic is implemented in a controller (i.e. the control plane). This technology makes it much easier to change the network programmatically, allowing network-wide traffic forwarding decisions to be implemented and network-wide monitoring and control of Quality of Service (QoS) simpler.

Network experiments requiring testbed support are by their nature at a pre-commercial stage. In the 1960's [1] proposed that adopters of any new innovation or idea could be categorized as innovators, early adopters, early majority, late majority and laggards. This rate of adoption has subsequently become known as the 'diffusion of innovations model', and represents an s-shaped curve when plotted over time. Different experimental testbed facilities support experimenters at different innovation stages. For innovators there are experimental testbeds such as the OFELIA experimental facility [6]. For early adopters and the early majority there are pre-commercial facilities such as BonFIRE [4] and federated testbeds made up of FIWARE Lab instances created under the XIFI project [5]. For early majority and late majority there are commercial cloud providers such as Microsoft Azure and Amazon EC2 [1]. Each testbed environment has its own particular set of characteristics, including its capabilities, limitations and support levels for practitioners running network experiments.

This paper reports our work on the qualitative evaluation of the practical limitations experienced when running network experiments across multiple experimental frameworks designed to support different innovation stages. We apply an idealized common experiment (connecting Minecraft[2] clients with a Minecraft server) to 3 different test beds, each designed for a different innovation stage with a focus on the practical experience of executing the experiment on each class and the limitations imposed on the experiment by the testbed. This experiment measures the network quality of service differentiated between premium and non-premium players. The hypothesis is that control of the data centre network topology allows bandwidth allocation to be guaranteed for premium users, even at peak load periods, which ultimately helps ensure a good Quality of Experience (QoE) for the Minecraft players. This experiment represents an archetype for experiments that network researchers might want to run on an experimental testbed facility.

II. RELATED WORK

Software-defined networking (SDN) [2] approaches are currently a very active network research area. The ease by which changes in network routing can now be executed is allowing new ideas to be implemented on testbeds quickly, helping to speeding up progress in the field of network research. In any SDN deployment there is a *northbound* interface, which allows applications to task the controller and set network policies, and a *southbound* interface, which

*The work presented in this paper is part of the research in the EU FP7 OFERTIE project (contract number 318665) and EU FP7 XIFI project (contract number 604590). In addition we would like to acknowledge the help provided by Jack Edge & Martin Hall-May (IT Innovation Centre) and Giuseppe Cossu (Create-Net) when setting up and running experiments.

[1] http://azure.microsoft.com; http://aws.amazon.com/ec2
[2] http://minecraft.net

978-1-4799-7899-1/15/$31.00 ©2015 IEEE

concerns the protocols used by the controller when programming network switches. OpenFlow[3] is one of the most common southbound interfaces, with many network switch vendors supporting it. The northbound interface is less well standardized [8] with a variety of application specific approaches on offer.

Network experimenters are applying SDN techniques to areas [9] [10] such as data centre management, internet service provider backbones, wireless networks and enterprise networks. Challenges for the experimenters include performance, scalability and resilience testing of SDN approaches. To support such experimenters a growing number of testbed facilities have been created, including programmes such as FIRE [7], the Future Internet [5] and commercial offerings such as Amazon EC2 and Microsoft Azure.

In the area of wireless networking there have been surveys [11] of the currently available testbed infrastructures, categorizing them using metrics. We want to do something similar for SDN wired network testbed facilities, reporting qualitative metrics based on our own practical experience using some of these testbeds. Our qualitative metrics categorized testbeds in terms of the features they provide and the support they offer for network experimenters.

III. Architecture

The architecture of the services that underpin the common experiment used in this paper connects an application server, in this case a Minecraft server, with its clients via a configurable network switch topology hosted by a data centre. The network switch topology is controlled by a SDN (typically via Floodlight [4] Openflow controller). A packet sniffer (e.g. TCPDump[5]) is used to record all network packet headers. We also added a plugin to the Minecraft server to publish ping statistics, providing us with a ground truth in-game latency measurement. Our network profiler is based on a supervised learning approach which can be trained to identify network events such as periods of high latency or packet loss and label them in our sample measurement database. Finally the network measurement and network behaviour profile data are checked against QoS guarantee thresholds appearing in a service level agreement to see if any thresholds have been violated and a penalty clause triggered for poor performance.

IV. Approach and Experiment Setup

We focus on measuring how testbed features limit the ability to perform an idealized experiment, and how effectively that experiment can be executed using the testbed support apparatus provided. Our experimental approach is to define an 'idealized' common network experiment which makes use of the architecture and components identified in section III. We then apply this common experiment to our 3 testbeds, each of which are tailored to support a different innovation stage. A set of qualitative metrics are then measured, recording the practical experiences whilst executing the experiment.

[3] https://www.opennetworking.org/sdn-resources/openflow

[4] http://www.projectfloodlight.org/floodlight

[5] http://www.tcpdump.org

Testbeds configuration is shown in Fig. 1 for (a) Amazon EC2 cloud, (b) FIWARE Lab instances and (c) OFELIA. We designed two versions of the experiment to support different capabilities. One version involving real players for Amazon EC2 and FIWARE lab instances and one involving bots and SDN control for FIWARE lab instances and OFELIA.

A. Common Experiment

The idealized common experiment run on each of the 3 testbeds consists of a Minecraft server hosting a mixture of premium and general users. A premium user is a user who has paid a subscription fee to receive better QoS than the free to play general user. The network topology of the testbed consists of a high bandwidth premium route and a low bandwidth best-efforts route to the Minecraft server. The premium player traffic is always routed through the high bandwidth path. Under low load conditions (i.e. phase 1) the general player traffic is routed across both the low and high bandwidth paths as the high bandwidth connection is nowhere near saturated. Under high load conditions (i.e. phase 2) only premium player traffic is routed through the high bandwidth path, leaving general player traffic using the best-efforts low bandwidth path.

TCPDump is run on the Minecraft server virtual machine so an independent measure of the network traffic is recorded, which is then parsed and profiled by the OFERTIE network services. The Minecraft server is a custom built Craft Bukkit [6]version of the Minecraft server with instrumented code to record a ground truth in-game latency measure for all players; this is used at the start of the experiment to execute a training run and create a Minecraft specific labelled training set for the supervised-learning-based classifier in the QoS profiler.

Halfway through each experiment the player load is increased from low to high for a short period of time; this is done by inviting many players to play on the server. We are interested to see how the SDN responds to the increase in player load and when it changes the network routing to ensure the premium players' bandwidth is always enough for a good QoE. The final part of the experiment involves SLA violation checks against an experiment WS-Agreement SLA document defining acceptable thresholds for QoS metrics (i.e. bandwidth and number/duration of classified high latency behaviour periods).

B. Qualitative Metrics

In order to evaluate our practical experiences trying to run the idealized experiment across each of the 3 testbeds we have defined a set of qualitative metrics. This approach allows a fair and consistent comparison to be performed across the otherwise heterogeneous testbeds. We use two types of metrics categorized into either feature support metrics or experimental support metrics. Below is a list of the qualitative metrics based on feature support and experimental support:

SDN support - we need programmatic switching of network routing in response to increases in network load. This can be either native support or via our own installed SDN software (e.g. OpenFlow).

[6] http://bukkit.org/

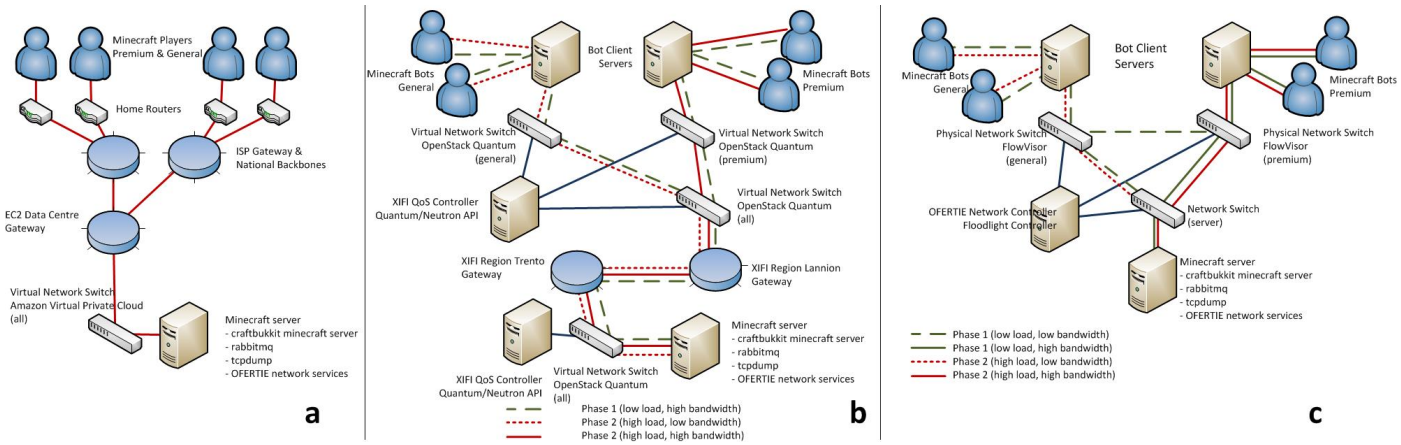

Fig. 1. Testbed configuration for (a) commercial Amazon EC2 cloud, (b) pre-commercial FIWARE Lab instances and (c) experimental OFELIA testbeds

QoS monitoring - we need to monitor real-time network level QoS (i.e. throughput/bandwidth, latency, packet loss) in order to base decisions about when to switch network routing. This can be via access to network switch counters (e.g. via Floodlight) or native measurement logs from the testbed itself.

External IP addresses - as we want real Minecraft players to connect we need the testbed to allow external IP addresses (i.e. players) to connect to it, and provide static Domain Name System (DNS) visible IP addresses so we can register the Minecraft server with publically available server listings to attract real players.

Network slice isolation - we need to isolate our network from other network traffic occurring on the testbed from other experiments. This is critical as we will be measuring patterns in network traffic and trying to do behaviour classification and SLA violation checking against it.

Reliability - we need a testbed that provides a reliable network slice that is stable for the several days each experiment will last. Major peaks in network performance or downtime will provide noise to the experimental results and probably render them useless for analysis.

Ease of access to testbed slice - we need easy remote access to the Minecraft server via standard tools such as Secure Shell (SSH). The more difficult and unreliable access is to the server the more time consuming experiment testbed to provide sufficient network and server resource to run the experiment.

Testbed helpline support - when problems do happen we need a fast and effective response to solve the issue and resume the experiment. Support is needed in both the experimental setup phase and runtime phase.

Cost - the cost of testbed provision must be affordable in relation to the benefits gained from running the experiment in the first place.

V. Results

A. *Experiment 1 - Commercial Class Testbed*

The Amazon EC2 cloud testbed was very simple to purchase and setup with good support via the Amazon EC2 web interface. The dynamic IP mapping provided by Amazon meant we could have as many external IP addresses as we needed. Amazon EC2 provides no support for SDN so we had to limit our experiment to a simple static network topology with no bandwidth control. Instead we focussed on profiling network high latency events from players with poor ISP connections as a replacement for being able to change the bandwidth in response to observed high throughput. We ran the experiment over several days and had up to 7 player online at any one time using our server. For QoS monitoring Amazon provides high level network usage reports, focussed on metrics for billing purposes. We therefore relied on the TCPDump data for our network monitoring purposes. We found the network slice isolation good, with the network connection never reaching a bandwidth limited situation. Overall it was easy to setup, 100% reliable and had a reasonable cost.

B. *Experiment 2 - Pre-commercial Class Testbed*

The XIFI project operates a 'cloud data centre' approach to networking similar to Amazon EC2, where network traffic between different tenants is globally optimized. As such this federated testbed does not provide direct access for tenants to the network switches or the routing tables of these switches. This choice to optimize globally across the supports testbed administrators, but prevents network experimenters (i.e. tenants) from running a wide range of experiment types that require access to switch measurement and routing.

For the first experiment, using real players, our observed behaviour is consistent with what we saw in the Amazon EC2 testbed. The experience in setting up this environment has been smooth and the support received adequate despite the need to synchronize our test period with hardware and testbed software upgrades that caused testbed downtime periods.

For the second experiment we used a manual interface to the XIFI controller to be able to set the QoS of the traffic directed or coming from each VM at run-time. Support for QoS management on incoming traffic based on the source of traffic is a feature not yet generally available in the OpenStack implementation, where QoS management is mostly applied for load balancing of the destination servers. The XIFI controller was therefore used to control QoS via soft throttling of bandwidth into and out of each VM. We simulated premium and general class users with groups of Minecraft bots assigned

to a premium or general class hosting VM, both in the Lannion testbed region. The Minecraft server was on another VM in the Trento testbed region.

Our practical experience in doing this was that using the QoS environment was challenging, despite receiving full support from the testbed administrators in the XIFI project. The QoS features therefore, although promising in nature, lack maturity meaning that experimenters must rely on ad-hoc support when running the experiments for now.

C. Experiment 3 - Experimental Class Testbed

The OFELIA testbeds allow access to the network layer via FlowVisor. However, this was not easy to set up and several problems occurred. The main problem with the OFELIA testbed was the unreliable behaviour of the network switches when working with the OFERTIE network controller. Switch downtime occurred frequently and each time took hours to resolve as access to switch reset could only be achieved via the testbed administrators.

In addition to switch problems the OFELIA testbed was difficult to access, requiring experimenters to pass through multiple VPN connections to get to the testbed. We also had to use Minecraft bots, as opposed to real players, as OFELIA islands were only set up to permit experiment connections to other OFELIA islands and not external ISP's. This is a serious limitation for any network experiment that involves real applications with real users.

The OFELIA facilities are free to use for experimenters but support is largely dependent on the goodwill of the testbed owners as there was no sustainable commercial funding stream in place on which to base support activities. Our experiments spanned a period at the end of, and beyond the end of, the OFELIA project's lifetime. Our experience is that the upgrades and maintenance schedules of the testbeds appeared to be correlated with the testbed owner's ability to participate in funded projects that use the testbed. This experience matches other FIRE testbed sustainability plans, such as AmpliFIRE and Fed4FIRE, where securing funding is linked to the continued performance of these test facilities.

After several months trying to overcome the network switching issues we eventually resorted to running the experiment on a Mininet emulation of the OFELIA testbed using manual SDN control. We could execute our experiment using bots connected to the testbed simulation which allowed us to test bandwidth control. We added network latency and packet loss to the Minecraft server using the NETEM[7] tool to allow us to test high latency profiling. However as this is essentially a simulation our empirical results were limited to supporting the validation of the OFERTIE network services.

VI. Discussion

A comparison matrix is shown in Table 1 highlighting the qualitative metrics across all 3 testbeds. Each testbed class offers a different range of capabilities. The practical issues experienced in running our experiments fall into four broad classes - testbed availability & reliability, aging testbed software, network slice isolation and access to external networks.

The issues of both testbed availability & reliability and aging testbed software are really related to the testbed owner's ability to maintain the testbed and have an effective incremental update strategy in place for both the hardware and software on that testbed. In contrast to commercial testbeds, where testbed use generates revenue to feed into testbed updates, we see that for pre-commercial and experimental class testbeds there is a reliance on external funding to be secured before upgrades are made. This leads to an ad-hoc incremental improvement strategy, where the testbeds have aging hardware and software for prolonged periods of time.

The issue of access to external networks creates restrictions on the ambition and value of the type of network experiments that can be executed. A lot of potential network experiments involve working with realistic network traffic, stochastic in nature and often based on real users using real applications over the network topology. This is hard to simulate accurately. Currently experimental class, and even to some degree pre-commercial testbeds in their current state, do not support this type of experiment well at all. It should also be noted that if a testbed cannot connect to an external network resource then the scale of the experiment is limited to the size of the testbed. For experimental class testbeds the scale tends to be small with 10's of CPU cores available for VM's and 5-10 network switches. For pre-commercial testbeds the scale is much larger with 1,000's of CPU cores available for VM's. Ambitious network experiments, such as trying to simulate part of a national network backbone topology, involve 10's to 100's of network switches. Currently experimental class testbeds are not large enough to allow this.

Network slice isolation is an important feature for most network experiments, since the presence of network traffic noise from other experimenters sharing a common network resource adds bias and/or error to the final measurements. We can see that commercial and pre-commercial class testbeds do not provide direct allocation of network switches to slices, and instead provide a virtual switch layered across an undeclared switch topology within a data centre. Typically bandwidth is soft throttled for tenants on these testbeds and everything else is on a best efforts basis. This can result in unpredictable packet loss and switch latency spikes when parallel experiments happen to reach peak load at the same time on the underlying physical switch. In practice this means that only experimental class testbeds can really support network experiments that want to work at the limits of a network switch's capability.

VII. Conclusions

We have reported in this paper our practical experience running network experiments across three classes of testbed facility. The commercial class testbeds (e.g. Amazon EC2) provide a standard cloud-based virtualized offering without SDN, and are well suited to large scale experiments that do not need to change the network topology.

[7] http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

TABLE 1: COMPARISON MATRIX FOR QUALITATIVE METRICS ACROSS ALL TESTBEDS

Qualitative Metric	Commercial Testbed Amazon EC2	Pre-Commercial Testbed Federated testbed of FIWARE LAB instances	Experimental Testbed OFELIA
SDN support	NO	YES in terms of QoS (bandwidth setup)	YES - outdated FlowVisor software prevented its use however in practice
QoS monitoring	Only network usage logging per VM is provided	Currently available for testbed owners and not for tenants but this is foreseen in the future	Access to switches was permitted via Floodlight controller
External IP addresses	YES	YES - although the QoS management enabled environment is currently private there are plans for it to go public soon	NO - testbed islands were only designed to connect to other islands
Network slice isolation	VPN is provided that soft throttles the underlying network bandwidth allocation.	Tenant isolation is guaranteed (run-time bandwidth) whilst complete network slice isolation is not (due to global optimization layer)	YES - network slices are provided
Reliability	Excellent	Fair	Poor - experiments are provided 'best efforts' support only and availability is intermittent.
Ease of access	Excellent - SSH credentials are provided for a direct SSH tunnel to VM	Good	Poor - multiple VPN layer are required to access slice from a remote location.
Scale of resources available	Excellent	Good - the experiment can transparently take advantage of inter-region MD-VPN	Average - compute and network hardware is heterogeneous and somewhat outdated.
Support for experimenters	Excellent - web interface provided to tenant	Excellent - web interface provided to tenant	Average - testbed administrators required for all restarts
Cost	Commercial rates	Free for FI-PPP experimenters	Free

The pre-commercial class testbeds (e.g. federated FIWARE Lab instances) also provide a virtualized network environment, but do support testbed-mediated SDN control of the underlying network. This is well suited for medium-scale network experiments in areas such as QoS management, where the network switches are not directly accessed. The experimental class testbeds (e.g. OFELIA facility) offer network slice isolation and direct access to network switches, supporting well small scale experiments that aim to stress network switch topologies.

At present no testbed class fully supports the range of network experiments researchers are working on today. The experimental class testbeds follow a closed 'sandbox' type approach, which prevent experiments from scaling up and does not allow realistic network traffic which usually comes from real users connecting via external ISP's. The pre-commercial and commercial class testbeds follow a scalable 'cloud data centre' type approach, where network resource is virtualized and therefore true network isolation is impossible. We think that there is a clear opportunity in the future for pre-commercial class testbed facilities to offer something different from commercial cloud offerings. They could use modern SDN controllers to provide experimenters with closed network switch topologies, and direct dedicated access to the network switches, alongside access to gateway switches offering external IP addresses and access to ISP network backbones.

If adopted this recommended change for pre-commercial class testbeds could significantly expand the type, scale and ambition of experiments possible on these testbeds. In turn this would increase the significance and value of the experimental results coming from the testbeds, which then leads to more measureable impact from the research activities supported by these testbed facilities.

REFERENCES

[1] E.M. Rogers, "Diffusion of Innovations", Glencoe: Free Press, ISBN 0-612-62843-4,1962

[2] H. Kim, N. Feamster, "Improving Network Management with Software Defined Networking", IEEE Communications Magazine, Feb 2013

[3] X. Jeannin, K. Meyer, "White Paper: The Opportunities for MultiDomain VPN services in GÉANT", GÉANT project, 2014

[4] A.C. Hume. et al. "Bonfire: A multi-cloud test facility for internet of services experimentation.", In Testbeds and Research Infrastructure. Development of Networks and Communities, pp. 81-96. Springer Berlin Heidelberg, 2012

[5] E. Escalona, et al. "Using SDN for cloud services provisioning: the XIFI use-case", In 2013 IEEE SDN for Future Networks and Services (SDN4FNS), pp. 1-7, Trento, 2013

[6] M. Suñé et al. "Design and implementation of the OFELIA FP7 facility: the European OpenFlow testbed", Computer Networks 61, 132-150, 2014

[7] S. Rao, "Research Experiment Facilities in Europe", White Paper, AmpliFIRE Support Action, 2013

[8] T. Humernbrum, F. Glinka, S. Gorlatch, “Using Software-Defined Networking for Real-Time Internet Applications”, In Proc. of the International MultiConference of Engineers and Computer Scientists (IMECS 2014), Lecture Notes in Engineering and Computer Science, Newswood Limited, pp. 150–155, 2014

[9] C. Kachris, K. Kanonakis, I. Tomkos, "Optical interconnection networks in data centers: recent trends and future challenges", IEEE Communications Magazine, Vol 51, Issue 9, 2013

[10] B.A.A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks", IEEE Communications Surveys & Tutorials, Vol. 16, Issue 3, 2014

[11] J. Horneber, A. Hergenroder, "Survey on Testbeds and Experimentation Environments for Wireless Sensor Networks", IEEE Communications Surveys & Tutorials, Vol. 16 , Issue 4, 201