Developing a Traffic Classification Platform for Enterprise Networks with SDN: Experiences & Lessons Learned.

Bryan Ng *Member, IEEE* Matthew Hayes Winston K.G. Seah *Senior Member, IEEE*
School of Engineering & Computer Science
Victoria University of Wellington
Wellington, New Zealand

***Abstract*—Software Defined Networking (SDN) is an innovative approach to networking architecture that opens up avenues to create a whole new class of networking functionality. While data centre networks are steadily adopting the SDN approach with considerable success, other areas of networking such as network access control, load balancing and traffic classification remain nascent. Traffic classification in SDN is relatively experimental and attempts for SDN traffic classification to become a viable solution for enterprise networks require additional investigation. This paper reports on the practical experiences and lessons learned while developing an SDN based traffic classification platform for an enterprise network. We use the platform to demonstrate the feasibility of SDN based traffic classifiers by evaluating against a set of desired outcomes. We make note of the design choices using the currently available technologies that may be helpful to networks operators considering deploying their own solution. We conclude the paper with suggested changes to better address limitations for software traffic classification that will remove the need for workarounds with future versions of OpenFlow.**

***Index Terms*—SDN, Traffic classification, QoS.**

I. Introduction

Traffic classification is the identification and linking of packet flows to traffic types. It serves as a foundation for a wide range of activities in networking, from network management to network security, from quality of service (QoS) to traffic engineering, from network analytics to big-data. In this context, the objects to classify are network traffic flows, which consist of sequences of packets exchanged between sets of endpoints, communicating over networks.

In this paper we are are concerned with traffic classification in enterprise networks. An enterprise network is a private network dedicated to carrying data communications for a single organisation traversing a public infrastructure. An enterprise network may have thousands, if not millions of packets in transit at any given moment, so it is a non-trivial exercise to identify the type of individual packets. The classification is based on different information of the traffic flows, such as port numbers, application payloads, and statistical features of the flows.

Traditional network traffic classification schemes, including port-based and payload-based classifications, rely on direct inspections of the network packets. For example port-based classification scheme inspects the packet headers, it examines the source and destination port number fields in the transport-layer (i.e., TCP and UDP) headers. The classifier identifies the application protocols according to the registered port number list maintained by the Internet Assigned Numbers Authority (IANA). Well known examples include port 80 is used for Web traffic (HTTP) and port 21 is used for file transfers (FTP). This approach is computationally efficient as it involves only simple access to the to the header level of packets and fixed-offset searches through a sorted integer list. However, it quickly becomes untenable for network administrators to write configurations matching per-port, per-device classification rules with the increasing diversity in enterprise networks supporting different devices, applications and capacity constraints.

While a broad range of academic papers cover traffic classification, there is a paucity of papers addressing the unique challenges of traffic classification in an enterprise environment, with the notable exception of [1], [2]. The lack of focus on enterprise networks from the academic community may be due to the closed nature of networking systems used by many enterprises, and commercial privacy concerns that prevent researchers from being able to get sharable traces [3]. Additionally, the mechanics of enterprise networks, where many parallel paths can exist, present challenges to obtaining representative packet captures, as noted in [4].

A. Challenges in enterprise networks

The advent of the Internet of Things (IoT) poses a growing challenge to effective traffic classification in enterprise networks. IoT is a fundamental change whereby a massive and diverse range of objects are becoming network addressable. This may be the networking of previously unconnected electronic devices in addition to proliferation of embedded networking functionality into previously non-networked items (e.g. signage, building structures, and clothing). It is estimated that the number of Internet-connected devices will grow from approximately 2.5 billion in 2010 to between 50 and 100 billion by 2020 [5].

Scalable and accurate traffic classification is a difficult problem because many enterprise network operators who are

ISBN 978-3-901882-68-5

interested in QoS do not know all the applications running on their network [6]. Port-based classifiers are increasingly out-of-favour with the advent of IoT because newer applications may not have a registered port number, while other applications deliberately hide traffic within well known port numbers. Moreover, with the trend of bring-you-own-device (BYOD) picking up, the number of networked devices in an enterprise will surely grow significantly as new uses are found for the services that they provide. A solution is required that at least partially automates traffic classification configuration so that organisations can efficiently and quickly apply and monitor traffic classification at a policy level, without having to make configurations on a per-flow, per-device or per-port basis.

B. Software defined networking

In this paper, the solution to traffic classification challenges in enterprise networks is premised on software defined networking (SDN). It separates the forwarding and control functions of network forwarding elements, making it possible to logically centralise control and apply a programmatic approach to the operation of a network. The programmatic approach facilitates the move from port and device based traffic classification to a less error prone approach such as the use of policy or higher level abstractions to deal with challenges in enterprise networks.

Since its release, the SDN scene has experienced substantial growth in the number of projects and is being extensively investigated for various network functionalities such as security, quality of service (QoS) etc. The logical centralisation of network control in SDN gives rise to innovation opportunities not afforded by discrete monolithic network architecture. For instance, it becomes possible to have a network–wide view of the network flow state. Network–wide awareness of flows in monolithic networks requires bespoke solutions and/or use of identifiers carried within or around packets. Examples of the latter include use of the differentiated services field in IP packet headers [7].

In the SDN architecture, the controller has a logically centralised view of the flow, removing the requirement to carry such administrative information in packets. This gives administrators the ability to write applications (or define policies) that leverage network–wide flow information and simultaneously creating a new space for innovation. SDN may also be able to assist with the prevailing problem in traffic classification for enterprise networks where privacy considerations prevent enterprise network traffic from being studied. A possible solution is to supply the analysis system to enterprises to run themselves, with only the resulting analytical data shipped back to the researchers [3]. This ensures that the research workers have no direct access to potentially private network data. SDN, where deployed on production networks, could allow researchers to construct systems that carry out traffic analysis without any requirement to install physical hardware. The system can be implemented as additional software on the SDN controller layer.

C. Contributions

This paper will discuss the development and deployment of a traffic classification platform using SDN at Victoria University of Wellington (VUW). The platform development was built with openly available SDN controllers and commercially available commodity hardware. To that end, this paper is a guide to network operators wanting to deploy traffic classification in SDN today and not a treatise on traffic classification algorithms or performance.

It should be clear that SDN is no panacea for the problems and challenges of today's networks. As a still developing technology, SDN has limited support for certain features, which may be desirable, for traffic classification in enterprise networks. Our first contribution is identifying the strengths of SDN for traffic classification and reflect upon the inertia of enterprise networks adopting SDN. For example, connecting remote switches with two separate links (control path and data path) per switch may not make economical sense for enterprises with thousands of remote switches.

Secondly, we show how to support traditional traffic classification based on SDN principles highlighting the advantages, challenges faced and lessons learned during the development of the traffic classification platform. We have found that SDN principles have varying degrees of benefits for different traffic classification methods. Identity classification was found to benefit immensely from centralised identity management conferred by SDN while static classification benefits least. A discussion on the extensions supporting traffic classification and how they may appear in future developments of SDN is given.

The remainder of this paper is organized as follows; Section II describes the key concepts underlying the traffic classification platform built on SDN and the considerations that went into the design. Section III presents results on the functional performance while Section IV details the lessons learned including security issues and the steps that can be taken to address them. Lastly section V provides a conclusion.

II. Software Defined Networking For Traffic Classification

Classifying traffic in a large corporate or enterprise network is a complex and time-consuming task. As the network consists of numerous discrete devices, each with multiple flows, it is clear that a more efficient and responsive framework should be made available to network operators. However, existing traffic classification (TC) mechanisms are closed and tightly bound to vendors implementations. Extensibility is next to zero as these devices are closed, propriety units that limit innovation.

Where many areas of computer services have evolved, the network has not – until now. The closed design of networking elements such as switches and routers is being challenged by OpenFlow [8], which is being increasingly accepted by both industry and academia. As an open specification, OpenFlow can be customised to analyse flow features in addition to specifying forwarding rules to switches. Therefore it is the

TABLE I
DESIGN CONSIDERATIONS FOR ENTERPRISE NETWORKS

Requirement	Description	Rationale
Selective determinism	Ability to set deterministic classifiers	Operators require consistent traffic classification behaviour for specified traffic types, so that actions (i.e. QoS treatment) can be performed on matching flows with a high degree of predictability.
Agility	Ability to classify unexpected traffic flows	A key tenet of the problem statement is that there are now too many flow types on the network for the operator to specify them all. Traffic classification must be able to intelligently classify unexpected flows.
Application awareness	Can classify dynamic flows based on knowledge of application behaviour	Can appropriately classify related flows started from an initial known protocol. Some applications start extra dynamic connections (i.e. NFS, SIP starts RTP, etc.).
Identity awareness	Support for classification based on endpoint identity	When devices were static it was relatively simple to write classification rules based on IP subnet/supernet as a surrogate for identity. With the proliferation of portable devices and wireless connectivity, IP addresses or subnets are no longer tied to a particular device and thus are not a good indicator of identity. For these reasons, operators desire a method to include other elements of identity in traffic classification rules.
Timeliness	Classifications are made within a short period of time, ideally before a large flow has had time to ramp up.	Timely classification is required for online consumers of traffic classification data, such as QoS and traffic engineering. There is no point applying QoS treatment to a flow if the classification data is not available until after the flow has finished.

most suitable enabler for demonstrating traffic classification in SDN.

A. Design considerations

Enterprise networks are heterogeneous; it is not possible to specify a standard example. The two points for consideration pertaining to traffic classification in enterprise networks are: (i) traffic classification requirements for enterprise networks and (ii) alignment of the requirements with the SDN paradigm. Requirements are thus surmised from common conditions that may exist in addition to the anecdotal experiences of the authors. We deduce that operators of enterprise networks are likely to have functional traffic classification requirements as per Table I.

Based on the rationale for the requirements, we argue for a policy-based traffic classification platform as it has advantages of flexibility and human-readability with no hindrance expressing the requirements set out in Table I. Other approaches we have considered for addressing enterprise requirements for traffic classification are rule-based approaches [9], [10] and ontology-based approaches [11], [12] which have merits in other domains of application.

B. Policy based classification for Enterprise networks

The system we are developing codenamed "**nmeta**"[13] (network metadata) leverages SDN by classifying new flows at the SDN controller layer, thus exploiting the distributed processing power and software flexibility that SDN affords. With the flexibility and programmability through SDN, we have decided to perform classification at the controller and the classifiers may be a combination of both software classifiers and hardware classifiers. Once a flow is classified, flow entries are installed to the forwarding tables (of individual switches) for efficient switching without further recourse to the controller.

This reactive approach that we adopt facilitates network–wide flow visibility and application of policy to classifier configuration. The nmeta platform supports multiple classification methods, and can thus be described as a multiclassifier. Classifiers can be specific (return a Boolean describing whether or not a match is made) or general (return parameters describing what they classified). Classifiers may be combined in a logical structure through use of a policy.

C. Modules for Traffic classifier

Nmeta employs a modular design, which decomposes major tasks into separate modules, with public interfaces and hidden implementation. This standard software design principle improves maintainability of code, since changes within a module are less likely to have unforeseen consequences outside the module.

Components of nmeta are grouped into regions that share a common purpose. The nmeta Core region (refer to the orange shaded area in Figure 1) manages communications with switches (i.e. processing of packet-in and switch messages, adding flows etc.) via OpenFlow and handles incoming Representational State Transfer (REST) API calls via the Ryu Python Web Server Gateway Interface (WSGI) libraries. There is only one module in this region, *nmeta.py*, which reads in the main configuration file on initialisation and processes Packet-in messages sequentially through the **_packet_in_handler** function.

The Traffic Classification region (refer to the blue shaded area in Figure 1) classifies packets against a traffic classification policy and returns results to nmeta Core. The *tc_policy.py* module reads in a traffic classification policy on initialisation, evaluates incoming packets against the policy and sends them to the appropriate classifier module (if required). A sample snippet of the policy is shown in Figure 2, and as pointed out earlier in Section II-B, it is intuitive, readable and hides details of the classifier implementation. The sample snippet shows

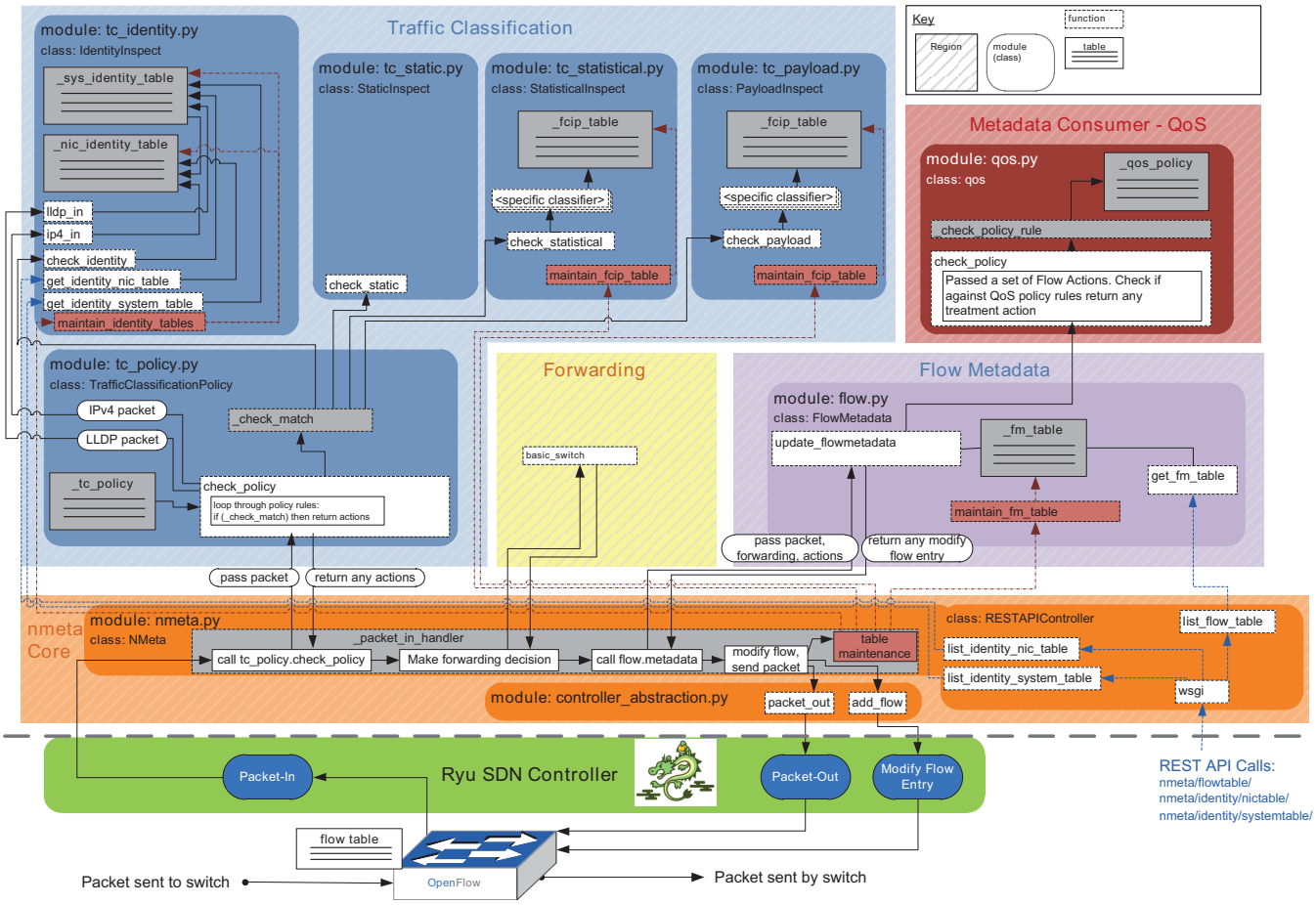


Fig. 1. Traffic classification architecture.

the specification of two policies: (i) an identity classification policy with matching conditions and corresponding actions and (ii) the statistical classification using linked policies ("PolicyRule 0" followed by "PolicyRule 1").

Currently, four classifier modules *tc_static.py, tc_identity.py, tc_payload.py* and *tc_statistical.py* contain the classification codes. The Flow Metadata region (refer to the purple shaded area in Figure 1) is called after forwarding decisions are made and communicated to the switches via OpenFlow messages.

The Flow Metadata region stores the enriched metadata in a Python data structure called a dictionary, and controls the installation of flow match entries to switches. The Metadata Consumer - QoS region (refer to the red shaded area in Figure 1) is a simple stub that provides a QoS treatment (queue assignment) based on matching a QoS flow metadata tag. Note that QoS treatment is not in scope for this paper so this region has been described as just the bare minimum required to run the test use cases in the next section. All communication from the traffic classification region to the flow metadata region is via the nmeta core region. This rule is to ensure that the forwarding module has visibility of traffic classification status messages.

D. OpenFlow

OpenFlow is a well-known protocol for establishing and maintaining control of the data plane. OpenFlow was chosen for the role of SDN protocol in the design due to its current popularity, large development community and non-proprietary nature. In the OpenFlow architecture, simple traffic classifiers, called "flow entries", are installed onto switches. A flow entry contains match fields which vary depending on the OpenFlow version. Where implemented in hardware, flow entry classifiers have the advantage of being relatively fast, but may have capacity and capability constraints [14]. As they occur within the data plane, their capabilities are dependent on the particular switch implementation, and they cannot directly leverage network knowledge outside of the switch view.

Hardware switch classifiers are relatively fast, but their capability is often limited due to constraints of the ASICs on which they are built. Software switch classifiers may be slower than their hardware equivalents, but are likely to have better feature support as their development is not dependant on support in silicon. Software classifiers are slower again due to the time taken to send packet(s) to the controller; however the benefits of software-development freedom, along with a network–wide view of flows, are judged to outweigh the

Test Identity-1 Policy

```
<snip>
'PolicyRule 3':
  comment: Use Case Identity-1 - High Priority Business Traffic
  match_type: any
  policy_conditions:
    identity_lldp_systemname_re: '.*\.audit\.example\.com'
  actions:
    set_qos_tag: QoS_treatment=high_priority
    set_desc_tag: description="High Priority Business Traffic"
```

Functional Test - Statistical Classifier

```
'PolicyRule 0':
  comment: SSH traffic
  match_type: any
  policy_conditions:
    tcp_src: 22
    tcp_dst: 22
  actions:
    set_qos_tag: QoS_treatment=high_priority
    set_desc_tag: description="High Priority SSH Traffic"
'PolicyRule 1':
  comment: Basic Statistical Classifier
  match_type: statistical
  policy_conditions:
    statistical_qos_bandwidth_1: on
  actions:
    pass_return_tags: true
```

Fig. 2. Snippets of policy specification in YAML.

performance downsides for traffic classification in enterprise networks. As mentioned earlier, all flows are classified initially by SDN controller classifiers and fine-grained classifiers are installed to switches once classification determinations are made.

III. Functional Evaluation & Performance Analysis

This section details the functionalities of two implemented classifiers and discusses a number of limitations encountered during the implementation. The discussion herein covers options for addressing these limitations and describes the impact on traffic classification. The functional evaluation is aimed at validating the classifiers with respect to desired outcomes as part of a software development life cycle. It is by no means a comprehensive evaluation of traffic classification in SDN. Tests were conducted on both OpenVswitch and commodity hardware, however, there was limited support for QoS functionality in the commodity hardware we are working with. Therefore we have decided to discuss the results from experiments using OpenVswitch. The discussion on comparing results from OpenvSwitch and hardware switches is deferred to the next section.

A. LLDP identity classification

The identity classification module records the identity of endpoints that broadcast Link Layer Discovery Protocol (LLDP) messages. Identity classification can be set to match against LLDP attributes values for example *chassisid* or *systemname*. The match can be a partial match defined as a regular expression or matching through Python's *in* operator. Identity information is stored in two dictionaries, one for Network Interface Controller (NIC) identities and the other for system identities. The system dictionary references entries in the NIC dictionary and vice versa.

Two dictionaries are required since an endpoint may have multiple network interface cards (NICs). LLDP Packet-in events are used by the identity module to accumulate system information and likewise, IPv4 Packet-in events are used to accumulate MAC address to IPv4 address linkages in the NIC dictionary. Matching against a *chassisid* or *systemname* value requires first checking if the value is present in the system dictionary. If present, the referenced NIC dictionary entry is retrieved and the packet is compared to see if it matches against the MAC or IPv4 values. If it does, a "True" value is returned otherwise a "False" value is returned and the classification is complete.

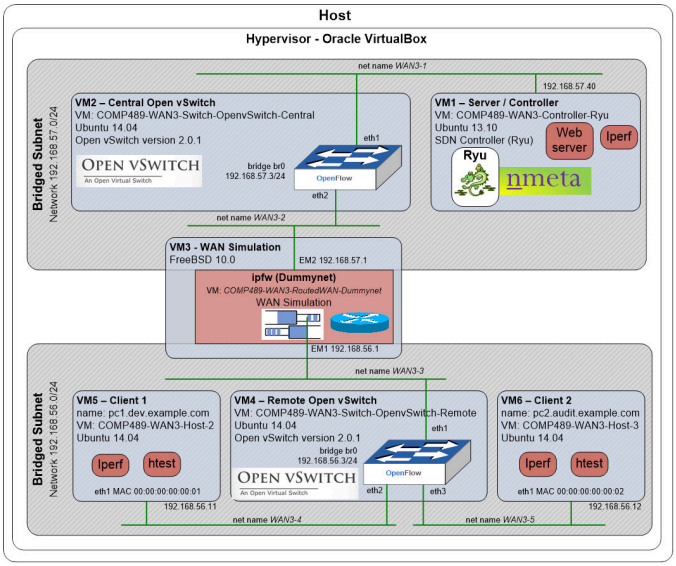


Fig. 3. Virtual lab topology.

1) Response times comparison: The test based on the scenario in Figure 3 is designed to demonstrate that the identity classifier can classify traffic to provide differential treatment of connectivity to/from a particular endpoint. Traffic classification is configured to treat as high priority any con-

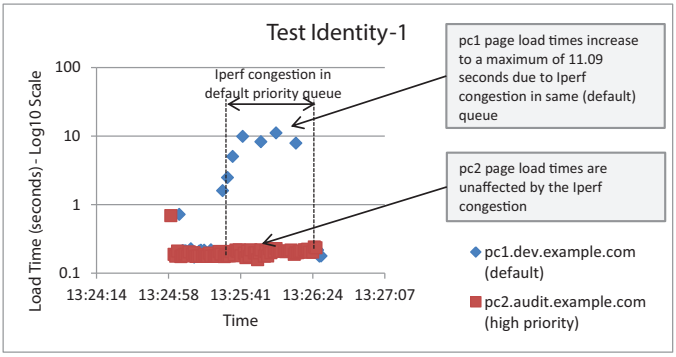


Fig. 4. Successful identity classification leads to lower response time.

nections to or from hosts that have an LLDP system name of **.audit.example.com*. Both **Client 1** and **Client 2** make regular HTTP connections to **Server / Controller** on tcp-80 and retrieve the same HTML object. Timing results are recorded for both **Client 1** and **Client 2**.

In order to demonstrate the functional performance with different traffic classifiers, it is necessary to estimate the baseline performance. In our tests, this is simply measuring the load time before ramping up the Iperf traffic generator and no classification policy is enforced. After establishing a baseline, Iperf from **Server / Controller** to **Client 1** and **Client 2** is used to congest the link in the default class for a sustained period. Iperf is then terminated and the test runs for a further period to recheck baseline. The desired outcome is – the load times of HTTP connections from **Client 2** (*pc2.audit.example.com*) to the server are not adversely affected by the Iperf congestion of the link.

Client 1 with LLDP system name *pc1.dev.example.com* is not matched by the identity classification. **Client 2** has an LLDP system name of *pc2.audit.example.com* and has its connections classified and treated as high priority based on the configured wildcard match for **.audit.example.com*. The response time to fetch the HTML object is shown in Figure 4. The base load time for both **Client 1** and **Client 2** is approximately 0.18s. As the Iperf congestion builds up, the load time for **Client 2** significantly increases while the load time for **Client 1** remains unaffected. Upon terminating Iperf, the load times for both **Client 1** and **Client 2** revert to the baseline load time observed before congestion was introduced. The load time differentiation between **Client 1** and **Client 2** clearly shows the successful classification based on endpoint identity classification.

2) Discussion: For identity classification it is desirable to have the corresponding flows for the return traffic classified identical to the forward flow. However, flows are traditionally defined with directionality and the same occurs when OpenFlow flows are created unless packets are broadcasts or multicast. For identity classification, the endpoint identities do not change with the flow direction. This characteristic of identity classification presents an opportunity for the SDN based approach.

In our implementation, the use of wildcard match on identity helps operators dealing with scale issues for traffic classification. Traffic differentiation is applied in both directions on the matched flows, including on the switch not directly connected to the identified device. This simplification eliminates errors due to omissions in reverse path classifications and reduces the number of matching criteria an operator needs to specify. This ability to make a system wide determination and apply it to all elements on the traffic path is an advantage conferred by SDN.

B. Statistical classification

Nmeta defines a bi-directional TCP flow as a 4-tuple of *ip_a*, *ip_b*, *tcp_port_a*, *tcp_port_b*. Packets can be matched as a flow in either direction as long as the TCP port numbers pair correctly with the IP addresses. A data structure called the Flow Classification In Progress (FCIP) table (a Python dictionary) is used to store flow classification state. A `continue_to_inspect` flag is used to indicate to the flow module that it should not install a flow to the switch as more packets need to be observed. If a dynamic port number is observed, it is added to the FCIP table so that packets in the dynamic port flow will be classified accordingly.

With the groundwork ready, the statistical classification module uses the FCIP data structure and computes the flow statistics of an unknown flow and finally compares the statistics with a set of pre-computed base signatures (we used the L7-filter [15]). Where the programmatic approach of SDN stands out, is in the ability to program the classifier to return actions, rather than just a Boolean for a match. The ability to return actions is required to be able to indicate between multiple possible results, such as classifying a flow to one of n traffic types.

1) Response time comparison: Next we demonstrate the statistical classification in nmeta. Again, referring to the setup in Figure 3, **Client 2** makes regular HTTP connections to **Server / Controller** on tcp-80 and retrieves a HTML object. A second test identically configured minus the statistical classifier is conducted as a control and load times are recorded. This functional tests shows the capability to run flow inspection on a sequence of packets, extracting statistical information and comparing it against the L7-filter signature. Timing results are recorded. The desired outcome is for the statistical classifier to classify the Iperf traffic into the *low_priority* queue, based on its statistical behaviour, and thus the Iperf traffic cannot impact the HTTP traffic since they are in different queues.

From Figure 5a, the baseline is determined to be approximately 0.19s. When Iperf traffic from **Server / Controller** to **Client 2** congests the link for a sustained period, there is a clear increase in load times up to two orders of magnitude, as shown in Figure 5b. Subsequently when Iperf is terminated, the test runs for a further period to recheck baseline and indeed the load time returns to the initial baseline values.

2) Discussion: When dealing with enterprise network traffic, it is tempting to look to the large corpus of statistical classification to classify the increasingly varied traffic brought upon by new devices and applications. In most cases the use of advanced statistical techniques involves a tradeoff between accuracy, delay and power. However, recall that installation of a flow has latency and processing overheads. Any additional delays by the statistical classifier should be cautiously regulated to maintain performance and overall usability of the network.

For example, when a Packet-in event is triggered due to the lack of a matching flow, the SDN controller needs to process that request, classify the flow and then forward the new flow modification request to the switch. These actions delay the subsequent packets and there is an inclination to set a time-to-live (TTL) counter to zero, and therefore an infinite lifetime, with the intention of reducing the amount of interactions.

However, such workarounds may have negative consequences. In situations where a user's access is required temporarily or a situation whereby previously granted access permission may have been revoked, there is now the possibility of a large number of flow entries with infinite lifetimes that are no longer required. The switch flow tables and the FCIP tables would be bloated, adversely affecting the network performance.

Besides concerns on bloated flow tables, a security vulnerability may have been created if the device has been passed on to a different user, which is often the case with devices within an enterprise network. These situations highlights some of the security and flow table capacity consideration that need to be addressed.

IV. Lessons Learned

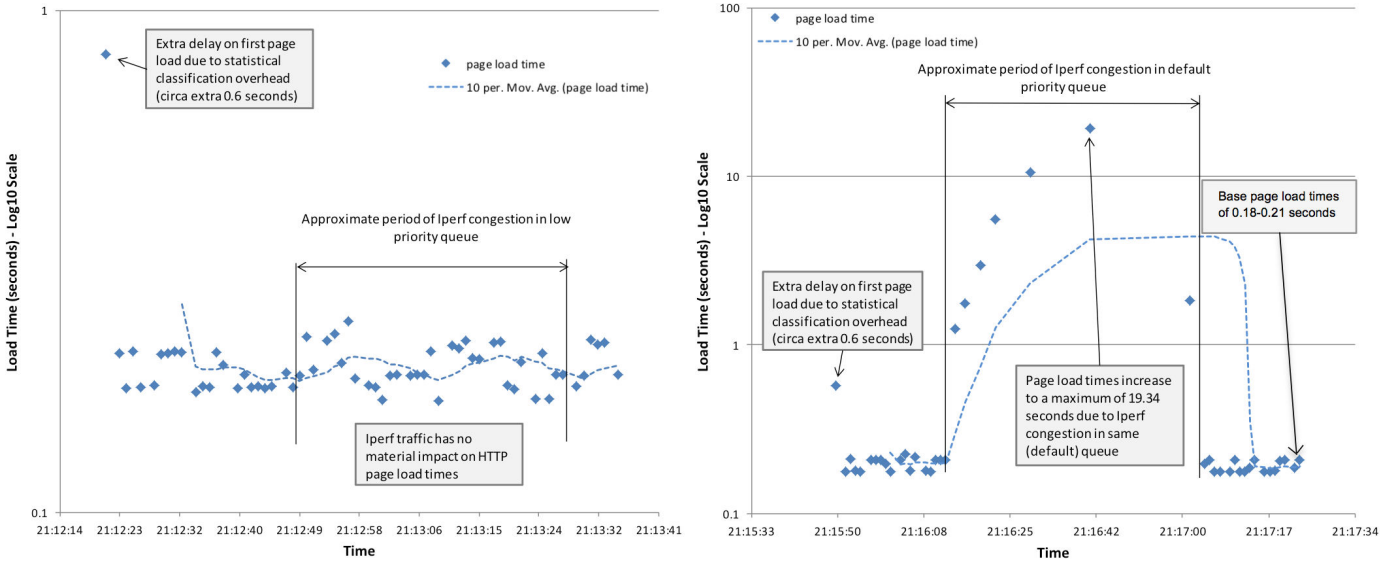
The efforts gone into the development of a traffic classification platform is continuing to provide valuable opportunities for researchers to experiment with SDN at VUW. While trial deployments of SDN are undergoing in isolated network segments, it would take several more months before any meaningful results can be obtained. We identified some limitations potential barriers and introduced a new approach to traffic classification using SDN for enterprise networks. This section describes two identified barriers for traffic classification and their implications.

A. Security

Enterprises take security seriously. It is unlikely that SDN will take hold in enterprises until it can be shown to be as secure as monolithic networking. It appears that SDN is lacking maturity in the area of security as exemplified by the discussion earlier in Section III. In nmeta, OpenFlow traffic is passed in plain text, which is great for troubleshooting, but not for security. This points to a critical need to look into the design of a safe control-plane interaction mechanism that allows traffic classifiers to maximise the flexibility of SDN while preserving classification correctness properties throughout the network.

Authentication through shared cryptographic keys such as the Group Secure Association Key Management Protocol (GSAKMP) [16] and the likes of it [17] are considered suitable protocols for providing secured communication for traffic classification modules. The salient point for GSAKMP is that it is capable of establishing secure associations between federated set of controllers and switches. GSAKMP enables secure and seamless key distribution with trusted authenticating network elements identified by groups. Each traffic classifier may require different interaction with the controller, and classifiers with common characteristics may be collectively addressed as a group. Thus, traffic classification modules may be viewed as groups communicating with another group or set of groups (a single controller or a set of controllers).

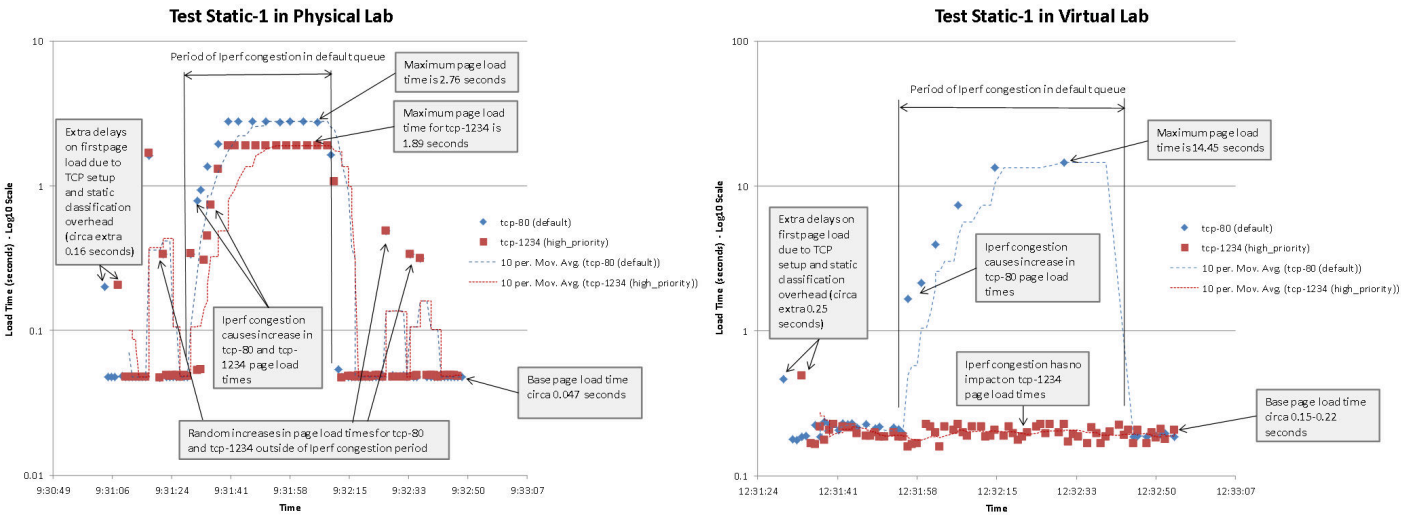
In our work with LLDP, it was found that LLDP packets passed to the Ryu controller with the default packet-in size of 128 bytes are truncated and causes Ryu to halt. This is a problem for both availability and security where it could be used to execute a Denial of Service (DoS) attack. While this problem is easily fixed using several lines of code, a better approach would be to enforce a policy regulating how traffic classifiers forward packets to the controller. We would caution would-be adopters to carefully examine this dimension of security because it is difficult to predict what sort interactions are expected of the plethora of SDN controllers. On a more specific note, our choice of using LLDP is purely for demonstrative purposes because LLDP is widely supported though we are well aware that it is not secure. Interested



(a) Page Load Time (Statistical Classifier)

(b) Page Load Time (Control)

Fig. 5. Response time with statistical classifier vs. control.



(a) Page Load Time (Physical Switch)

(b) Page Load Time (OpenvSwitch)

Fig. 6. Static classification response time in physical lab vs. Response time in virtual lab.

readers are referred to [18], [19] for recent issues concerning security in SDN.

B. Flow classification performance

Nmeta is a single threaded Ryu application and hence, it is susceptible to blocking. The REST API shares the same thread, potentially causing performance degradation if called on a large dictionary while the system is under load. There is an opportunity to improve performance for situations where classifiers need to see more than the first packet in a flow (i.e. payload and statistical) on flows that cross multiple switches.

The current nmeta behaviour is to require a packet-in from each switch in each direction until the classification has been made. Multiple switches result in duplicate packet-in events being sent to the controller that add no value. Nmeta is configured to ignore these duplicate packet-in events; however it is worth noting that they impact performance as they add load to the backhaul and controller. They also delay the forwarding of the packet until the controller has sent a packet-out message. If there are k switches and x packets must be observed then there will be $x(k-1)$ duplicate packet-in events. To improve this situation, the controller could install flow table entries to all but one of the in-path switches, and update these entries if required based on the traffic classification determination. Taking a cue from X.882 [20], we believe that equipping the northbound interface with support for remote operations is a step forward to enable efficient traffic classification among distributed SDN domains.

C. Hardware quirks

A physical lab was setup by substituting the OpenvSwitch with a commodity switch in the scenario shown in Figure 3. In test Static-1, the network is configured to classify and treat tcp-1234 as high priority and all other traffic as default. As in the virtual lab scenario, the desired outcomes are – load times for HTTP objects over tcp-1234 are not materially affected by the link congestion while the load times for HTTP objects over tcp-80 are noticeably affected by the congestion.

The load times in Figure 6 show that the initial load times (in each series) were higher due to overhead of TCP session establishment and flow classification via the SDN controller. The first tests took approximately 0.25 seconds in the virtual lab setting, whereas in the physical lab they took approximately 0.16 seconds. The overheads of virtualisation are likely to have contributed to the higher first test load time in the virtual lab when compared to the physical lab.

Response times for HTTP connections on tcp-1234 in test Static-1 in the virtual lab were not materially affected by the link congestion, meeting the expectations of desired outcome. Physical lab results indicate that the tcp-1234 traffic was correctly sent to the high priority queue, however the hardware queueing implementation on the commodity switches does not provide adequate isolation. Therefore the traffic in the high priority queue was impacted by the Iperf congestion to within 68% of the increase observed in the default priority queue.

The physical lab results (as shown in Figure 6a) highlights the importance of choosing SDN switch hardware carefully, and testing it to ensure that it meets requirements. It is not uncommon for the first generation of a hardware implementation to have limited features. As the OpenFlow standard matures, future hardware releases will be less restrictive and more consistent.

V. Conclusion

The traffic classification platform that we are developing is shown to meet the desired outcomes of traffic classification and shown to work with commodity hardware. Our efforts are helping to identify practical issues with the roll out of traffic classification in SDN. However, the work is far from complete as far as getting enterprise networks to adopt it. In the process,

we detected potential incompatibilities with legacy networking devices and protocols, and uncovered indications of possible implementation barriers for enterprise network adoption.

Shortcomings with the OpenFlow standard for traffic classification, such as, port range matching and TTL sizes are simply resolved with updates to the standard. However, other dimensions of the traffic classification problem in enterprise networks such as scalable classification, joint software-hardware classifier optimisation and secure controller-switch communication requires deeper thought. SDN is indeed radically changing the way network services are offered across different environments. It is however, in the process of maturing and hence still has significant room for advancement.

REFERENCES

[1] T. En-Najjary and G. Urvoy-Keller, "A first look at traffic classification in enterprise networks," in *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*. ACM, 2010, pp. 764–768.

[2] B. Ujcich, K.-C. Wang, B. Parker, and D. Schmiedt, "Thoughts on the Internet architecture from a modern enterprise network outage," in *Network Operations and Management Symposium (NOMS)*. IEEE, 2012, pp. 494–497.

[3] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *Network, IEEE*, vol. 26, no. 1, pp. 35–40, 2012.

[4] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney, "A first look at modern enterprise traffic," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, 2005, pp. 2–2.

[5] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, *Vision and challenges for realising the Internet of Things*. EUR-OP, 2010.

[6] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004, pp. 135–148.

[7] K. Nichols, D. L. Black, S. Blake, and F. Baker, "Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers," *RFC2474*, 1998.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[9] K. Appleby, S. Calo, J. Giles, and K.-W. Lee, "Policy-based automated provisioning," *IBM Systems Journal*, vol. 43, no. 1, pp. 121–135, 2004.

[10] C. Rensing, M. Karsten, and B. Stiller, "AAA: a survey and a policy-based architecture and framework," *IEEE Network*, vol. 16, no. 6, pp. 22–27, 2002.

[11] A. Bernstein, F. Provost, and S. Hill, "Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification," *IEEE Transactions on Knowledge and Data Engineering,*, vol. 17, no. 4, pp. 503–518, 2005.

[12] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services," *Information Technology and Management*, vol. 6, no. 1, pp. 17–39, 2005.

[13] "nmeta," https://github.com/mcprojectvuw/MC, accessed: 2014-12-07.

[14] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[15] J. Levandoski, E. Sommer, M. Strait *et al.*, "Application layer packet classifier for Linux," 2008. [Online]. Available: http://l7-filter.sourceforge.net/

[16] H. Harney, U. Meth, A. Colegrove, and G. Gross, "GSAKMP: Group secure association key management protocol," *RFC4535*, 2006.

[17] J. A. Cooley, R. I. Khazan, B. W. Fuller, and G. E. Pickard, "GROK: A practical system for securing group communications," in *Proceedings of the 9th IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, USA, 15-17 July 2010, pp. 100–107.

[18] K. Benton, L. J. Camp, and C. Small, "Openflow vulnerability assessment," in *Proceedings of the 2nd ACM SIGCOMM workshop on Hot topics in Software Defined Networking*, Hong Kong, 16 August 2013, pp. 151–152.

[19] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," in *Proceedings of 2013 Workshop on SDN for Future Networks and Services (SDN4FNS)*, Trento, Italy, 11-13 November 2013, pp. 1–7.

[20] ITU-T, "Remote operations service element (ROSE) protocol specification," *Interfaces*, vol. 10, no. 882-X, p. 49, July 1994.