PUCPLight: a SDN/OpenFlow Controller for an Academic Campus Network

Gabriel Cuba, Juan Manuel Becerra, Gumercindo Bartra, Cesar Santivanez
Advanced Networks Research Group (GIRA)
Pontificia Universidad Católica del Perú (PUCP), Lima - Perú
Email: {cuba.gabriel, a20090417, gbartra, csantivanez}@pucp.pe

***Abstract*— Campus networks – with his high diversity of users, traffic requirements, fast changing roles, and openness of equipment (BYOD) -- present multiple unique challenges difficult to meet with traditional networking technologies. SDN/OpenFlow provides a new toolset to rethink and redesign a campus network with aims at higher scalability, flexibility, and lower cost. This paper presents PUCPLight, an SDN/OpenFlow controller for a layer 2 campus network of tens of thousands of users. PUCPLight is a first approximation to a vision of a pure SDN scalable, flexible campus network that seamlessly supports from administrative services, to high performance computing, to experimental research. This paper describes the drives behind PUCPLight design, some of the design choices, as well as initial results proving scalability to a subnet of 10K nodes.**

Keywords—openflow; sdn; networking; controller; switch

I. INTRODUCTION

Academic campus networks, such as the PUCP's have a lot of users, with different roles, requirements and Quality of Experience (QoE); with increasing demands for information ranging from the use of educational platforms, email and social networks, to scientific/high performance computing and support of research platforms and videoconferencing. This diversity, together with inefficient bandwidth management of existing L2/L3 technologies that routes all packets over the same set of paths, leads to traffic congestion and degradation of the Quality of Service (QoS) experienced by the users. Thus, as network size and traffic grows, users change roles, and new – unaudited -- devices are added following the emerging Bring Your Own Device (BYOD) paradigm, the network becomes unstable and difficult to maintain.

SDN is a recent networking paradigm that removes the control plane (intelligence) from the network elements (switches, routers, etc.) and (logically) centralizes it in an element called Controller. The SDN Controller is aware of the entire network state and is better able to make efficient use of network resources, providing greater flexibility and scalability on the data plane. An SDN-based network has the potential for a reduced cost of operation and maintenance, and – since it is based on open standards, – can avoid vendor lock-in, allowing plug-and-play of devices in a multivendor environment.

This freeing of equipment vendor and their innovation/release cycles, allows a network engineer to design and implement new features as required by its institution. In this sense, SDN presents a unique opportunity for the development of the IT sector in Peru. Under the traditional paradigm, the relies in big equipment vendor that offer an integrated software-hardware solution, the country has been limited to be a passive consumer of technology, unable to compete with the economies of scale associated with hardware manufacturing, and becoming hostage to the vendor´s feature roadmaps and release cycles. With SDN and its decoupling of hardware and software, it is now possible to create a local telecommunications industry developing custom-made solutions that are implemented as a SDN/OpenFlow controller. This controller then in run is conjunction (controlling) a network of COTS equipment (OpenFlow switches, either white box [1], or high performance OpenFlow-enabled switches from known vendors such as Arista[2]).

The PUCP is undergoing a transformation, from a teaching university to a research one [3]. From a networking point of view, this drives new requirements for a rich internal switching fabric that support scientific computing and high speed access among collaborating research groups, as opposed to a network designed to shared traditional (relatively lower speed) access to internet and a small number of well-known web-based application servers. Flexible role-base access to shared resources (such as internet access) is also needed.

The aim of this work is to design and implement an SDN controller for an academic campus network, taking as reference the scenario and requirements of the PUCP present and future campus network, so that this is scalable to large traffic demands and diversity of user requirements, as well reliable, and flexible. The design is modular, and it is implementation is based on the platform provided by the Floodlight controller [4], to which specific modules are. The design considers a gradual implementation, allowing the coexistence islands of SDN/OpenFlow and Legacy network equipment.

This work is, to the best of our knowledge, the first attempt to design a SDN network for a Peruvian campus from the ground up. It has been developed by PUCP´s Advanced Networks Research Group (GIRA, for its initials in Spanish), taking advantage of the group's SDN/OpenFlow testbed.

The rest of the paper is organized as follows: section II presents the drivers behind the design of PUCPLight, section III presents a brief description of PUCPLight, section IV presents the results of validation testing via simulation and on a testbed. Section V presents some performance gains over todays networks thanks to PUCPLight, and Finally Sections VI and VII show our conclusions and future work.

978-1-5090-2532-9/16/$31.00 ©2016 IEEE

II. DESIGN CONSIDERATIONS

A. *Open Source Software, programming language and controller*

The adoption of an Open Source model avoids vendor´s lock-in and facilitates system´s upgrades and modifications. The usage of Open Source hardware and software, with the proper design, ensures interoperability between the different components inside the network.

In the choice of programming language, a crucial requirement for Core and Controller Applications was high performance. It is shown in [7] that Java-based controllers have better performance compared to Python-based controller and similar to a C-based controller. However, for ease of implementation and rapid deployment of REST applications, Python has an edge over the others. Thus, a more detailed analysis is required.

There are several studies comparing different open source controller platforms that meet our requirements. In [6], the authors compare NOX, Beacon, Maestro and Beacon-MT, using a set of tools called Cbench. CBench simulates a large number of switches sending "Packet Ins" to each controller. It concludes that Beacon-MT presents better performance due to its multi-threading capabilities. In [7], the authors – in addition to CBench -- use the HCPROBE tool to evaluate the reliability and safety of the controllers, concluding that Beacon supports larger throughputs. Finally, the study in [8] uses a Multi-Criteria Decision Making method, called Analytic Hierarchy Process (AHP), which allows to qualitatively compare the controllers' properties and to determine the best one based on different priorities. In this study, the best controller was Ryu followed by Floodlight.

Based on these studies, Floodlight was determined to be the platform that best balances performance with easy of (rapid) development, and was chosen for the implementation of PUCPLight's proof-of-concept.

B. *Campus network traffic requirements*

In order to characterize the traffic, prior to the design, some traffic captures were made to observe the behavior of a "typical" user within the campus network (an undergraduate student) and found out that the most of the traffic is directed to the Default Gateway to Internet. The second type of traffic observed in volume was Broadcast traffic due to the current VLAN segmentation, which is intended to be diminished.

C. *Unicast traffic and Internet requirements*

Unicast traffic represents the largest volume of traffic in the network and it is seen that the destinations' distribution is not uniform. Upon reviewing of the school core switch's traffic reports, it could be noted that a small number of hosts were the destination of most of the recorded flows. This observation was consistent with previous reports that Internet and data centers traffic show a Self-similar behavior (with respect to destination, duration, and volume), and can be modeled accurately by Pareto distributions [5]. It was also consistent with the 80-20 rule (20% of users are responsible for 80% of traffic), which will be used for our design and scalability analysis.

D. *Unicast Routing*

Our base mechanism for unicast traffic performs a Shortest Path, destination-based routing as a way to achieve load balancing, higher throughput, and scalability. A two-level Clustering system that performs tag-based routing (similar to MPLS and SPB) has been implemented. The use of tags/labels allows hosts to be grouped per Access Switch to decrease the number of rules needed at the distribution and core switches.

III. PUCPLIGHT DESCRIPTION

As mentioned before, PUCPLight is developed over the base of the FloodLight platform. Figure 1 shows PUCPLight software modules and their relationship to Floodlight modules. Floodlight modules that were left untouched are painted skyblue. Modules that were imported from existing internet libraries are shown in gray. Finally, modules that were implemented from scratch are shown in green. This section very briefly describes PUCPLight modules and operation:

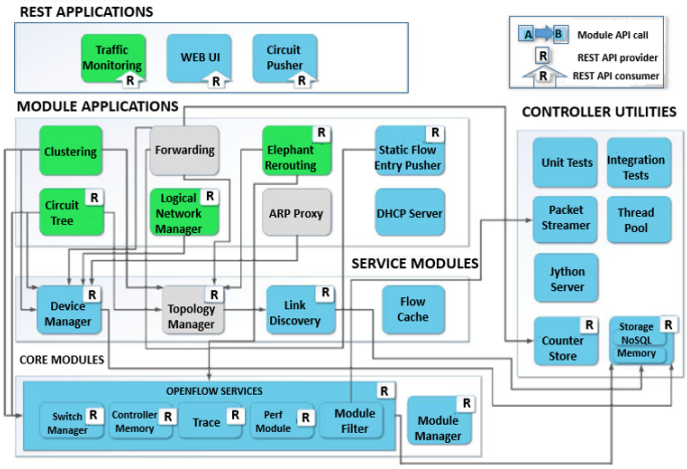

Figure 1. PUCPLight software modules.

The network is considered a single subnet. That is, hosts can talk directly without an intermediate router. Broadcasts, however, will receive special treatment and never traverse the entire network.

The *Logical Network Manager* module provides a logical segmentation of the network. PUCPLight allows the existence of multiple instances of the network and every host is allowed to belong to one or more of them. This module, by default, allows traffic to flow between instances, but it also permits the blocking of it as solicited for the administrator, or the creation of totally isolated independent instances.

The default *Forwarding* module will be extended with a two-level hierarchical routing mechanism (Clustering). Typically, a cluster will be formed by an access switch (cluster head) and the hosts it is serving, although bigger clusters are possible. The level-2 topology is formed by the aforementioned cluster heads (i.e., access switches) as well as

all the higher tier switches (distribution, core, etc.). PUCLight implements proactive Shortest Path Forwarding at the level-2 topology (via high-priority rules at the switches). At the level-1 topology, flow entries matching the destination host MAC address to the level-2 address (tag) are dynamically set by the controller in response to new session arrivals. These flow entries will have a relatively short) expiration timeout. Once a flow entry is set up, the data plane behavior for unicast traffic is similar to Shortest Path Bridging.

The *Circuit Tree* module determines the flow entries for the level-2 topology. For each cluster head (Access Switch) it creates a tree (named the 'circuit root') from this cluster head to every other cluster head. Then, for each switch in the tree, this module adds an entry indicating how to forward packets destined at level 2) to the root of the tree.

Broadcast off -loading is achieved by the use of the *ARP Proxy* and *DHCP Handler* modules that converts most broadcasts into unicast packets sent to the controller. The rest of the broadcasts are multicasted to the "Communities of Interest (CoIs)" the sender belongs to. A CoI is a group of hosts that share a broadcast domain (for example, to enable service discovery) and in that sense is similar to today´s VLANs. However, as opposed to today´s VLANs, a node can belong to multiple CoIs through the same interface and without the need to add tags. It should be noted that since switches generally do not need to be part of a CoI, they no longer need to process broadcast packets – freeing their CPUs. This is particularly important for the typical level-3 switch at the core of the network, which currently has to process every single broadcast packet sent in the every subnet.

The building of a CoI and tracking of its hosts, is performed by the *Logical Network Manager* module. It creates a multicast distribution tree that minimizes the number of transmissions (number of links in the tree) through a heuristic that provides an efficient outcome, in a short time, of the Minimum Steiner Tree problem, which is proved to be NP-Complete. A distribution tree is implemented using OpenFlow groups with a special mechanisms added to deal with packets bouncing back an edge of the Tree.

Additionally, the Elephant Rerouting module will be used -- as a first step of implementing Traffic Engineering -- to re-route the heavy flows determined to be elephants by the *Traffic Monitoring* and *Elephant Detector* modules, according to a calculation and assignation of routes with a dynamic metric (which for now is links utilization).

IV. TESTING AND VALIDATION

Different experiments were conducted to tests the validity of the design and its scalability limits. This section describes some of them that show that PUCPLight can scale up and above 10K nodes.

- **PUCPLight validation test over simulated network.**

A 500-hosts scenario, shown in Figure 2, was used to test the design as well as the proper behavior of PUCPLight. The network in this scenario has a hierarchical structure, consisting of 4 Core switches, 20 Distribution switches, and 100 Access switches. The 4 Core switches are connected in a full-mesh topology. In addition, each Core switch serves 5 Distribution switches, which serve 5 Access switches each one, and so on. This topology was simulated using Mininet [9].

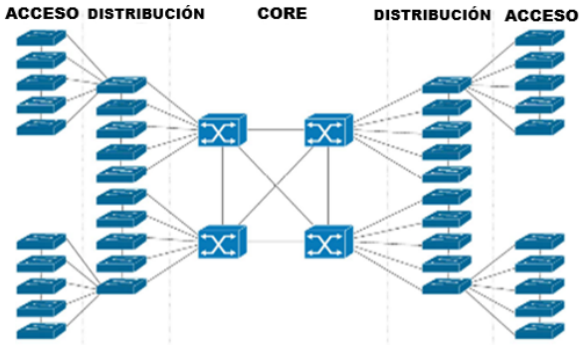

Figure 2. 500-host simulated network

First, it was verified that PUCPLight is able to detect the topology and create shortest paths between the different Access switches. Then, tests were made to validate the correct handling of broadcast packets. With the help of Wireshark tapping an interface on one of the Core switches, it was verified that PUCPLight captured and responded to ARP and DHCP requests. Specifically, ICMP ping packets were flooded through the network enabling and disabling the *ARP Proxy* and *DHCP Handler* modules. In the first case (with modules disabled) many ARP packets were observed (almost 97% of the captured traffic). In the second case (modules enabled) the number of ARP packets observed was reduced to 10% of the captured packets, meaning that PUCPLight successfully intercepts and proxies them. In both cases, the pings were successful, meaning the system was working correctly.

- **PUCPLight stress test on hardware testbed**

In order to assess the correct operation on a real-life network and well as determine the number of requests that PUPCLight can handle when running on a COTS computer, the scenario shown in Figure 3 was deployed and tested, using real life hardware and SDN switches. The topology consists of 3 hosts, each connected to 1 Access switch (Pica 8 model 3297, 48 1GbE ports + 4 10GbE uplink), which in turn are connected to a single Core switch (Pica8 model 3922, 48 10GbE ports). A separate 1GbE control-plane network (not shown) connected the management port of each SDN Switch to a Dell Precision T3610 Quad-core workstation running PUCPLight through a Dell N3048 switch.

After the controller discovered the topology correctly and connectivity tests (ICMP pings) were successful, the following tests were successfully performed:

- Iperf tests to measure the bandwidth of the control-plane and user-plane links.
- Verification of the proper installation of the Clustering module rules.
- Web browsing on all hosts through a host configured as a Gateway to the Internet.

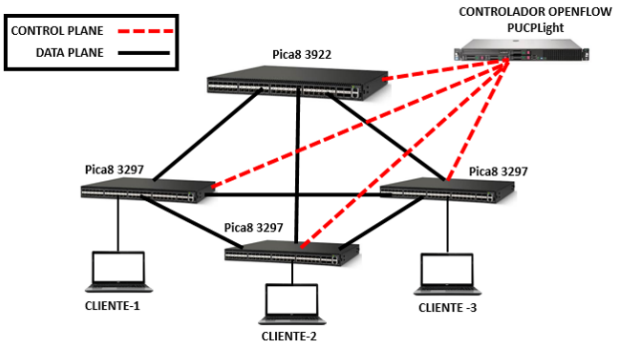

Figure 3. SDN Testbed

- Interoperability with legacy islands (in this case, the school network), through the recognition of MAC addresses of all network elements sending broadcast packets (mostly gratuitous ARP) to the PUCPLight network.

After these tests were successful, the tool CBench was used to generate a large amount of controller requests and measure the response capacity of the controller. These results are shown in Table 1. It can be seen that in average, PUCPLight running on a COTS computer can handle around 50K requests per second, which is more than enough for a 10K node network.

#switches	Modo Latency		
	min	max	avg
1	6 520.99	19 580.98	18 139.50
10	45 491.95	49 838.95	48 208.09
100	41 718.42	56 228.44	53 923.55
200	36 416.89	56 767.83	51 879.78

Table 1. Requests per second handled by controller.

V. Performance Assessment

Due to space constraints, a complete performance comparison for all possible scenarios is not possible. Instead, we summarize performance gains of PUCPLight with respect to today's PUCP network, where the higher tier is composed of a star topology with a core Layer-3 switch at the center connecting 40 buildings (approximately) via 1Gbps links. The core switch is the default gateway of all the school VLANs over 300) and performs all routing functionalities between them. PUCPLight improves the PUCP network by:

- **Better exploitation the spare capacity**. Traditional spanning tree approaches work by disabling spare links. Instead, PUCPLight can and will use those links to maximize throughput. For example, consider a fiber ring going around the 40 buildings to provide backup against fiber cuts. MST will not be able to exploit these links capacity. However, PUCPLight will use it to load balance traffic. Assuming that these links are 10Gbps (reasonably due to their short distance and the availability of 10GbE uplink ports unused in the distribution switches) and that the traffic is uniformly distributed among buildings, then PUCPLight will achieve a 300% increase in throughput per building.
- **Better scalability**, since not all traffic has to go through the core switch. Besides, since there is no need for the core switch to be member of any CoI, no broadcast packet (other than the infrequent ARP request coming from the controller's *ARP proxy* module) will have to be processed by the core switch. This switch CPU is highly loaded as it is. On the other hand, the controller can increase its processing power (i.e., cores) as needed.

VI. Conclusions

This paper introduces PUCPLight, a scalable solution to academic campus networks.

The use of a 2-level hierarchy allows for optimal routing and load balancing at the distribution and core network, resulting in higher throughput while at the same time reducing the processing load at the higher tier switches.

PUCPLight mechanisms of broadcast handling (ARP Proxy, DHCP Handler, and Community of Interests) have significantly reduced the broadcast traffic in the network, not just saving bandwidth but more importantly freeing valuable CPU cycles in overloaded core switches.

PUCPLight can coexist with elements of Legacy islands, and supports a total migration of the PUCP campus network to a fully OpenFlow/SDN topology.

VII. Future Work

PUCPLight has been envisioned to provide role-based network access to resources, as such, it needs to be integrated with existing AAA services (e.g., RADIUS), and the Enterprise WiFi network. Also, it should extend its ARP and DHCP handling support to IPv6. Finally, for High availability, an Active - Stand By controller scheme must be developed and strictly tested.

References

[1] Pica8 Inc. (10 de 12 de 2015). Pre-loaded Switches. Obtenido de Pica8: http://www.pica8.com/products/pre-loaded-switches

[2] https://www.arista.com/en/products/switches

[3] http://files.pucp.edu.pe/homepucp/uploads/2016/04/29104350/02-Plan-estrategico-institucional-2011-20171.pdf

[4] Floodlight Developers Mailing List. https://groups.google.com/a/openflowhub.org/forum/#!forum/floodlight-dev

[5] Smith, R. D. (2011). The dynamics of internet traffic: self-similarity, selforganization, and complex phenomena. Advances in Complex Systems,14(06), 905-949.

[6] Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., & Sherwood, R. (2012, April). On controller performance in software-defined networks.

[7] Shalimov, A., Zuikov, D., Zimarina, D., Pashkov, V., & Smeliansky, R. (2013, October). Advanced study of SDN/OpenFlow controllers. In Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia (p. 1). ACM.

[8] Khondoker, R., Zaalouk, A., Marx, R., & Bayarou, K. (2014, January). Feature-based comparison and selection of Software Defined Networking (SDN) controllers. In Computer Applications and Information Systems (WCCAIS), 2014 IEEE World Congress.

[9] www.mininet.org