QSIURP 2015: Report

SOFTWARE DEFINED NETWORKING IN WIRELESS NETWORKS

Muhammad Ahmed Shah

ABSTRACT

Wireless networks share a common channel that is spatially reused. This gives rise to interesting and complex interference phenomena that have substantial effects on performance and fairness. The shared channel makes wireless networks a strongly coupled system where decisions by one node can have a substantial and cascading effect on other nodes in the system. Developing effective distributed protocols in such an environment is challenging since local decisions can have significant effect making it difficult to converge to effective operating points.

SDN is establishing itself as a game changing technology in managing network infrastructure in a number of networking domains. SDNs makes it possible to carry out decisions that are globally effective and introduce the level of coordination not possible with distributed protocols. In addition, an SDN framework offers advantages with respect to service deployment, network monitoring and instrumentation as well as security.

In this research project, we review some of the recent developments in wireless mobility, investigate the possibility of predicting paths to pre-empt access point handoff and investigate the feasibility of using Raspberry PIs – a lowcost credit card-sized single-board computer, as an SDN enabled wireless routers for a Wireless Mesh Network.

INTRODUCTION

Software Defined Networking (SDN), a new architecture that brings software programmability to networks, is transforming the networking industry [4][5]. It aims to make networks easier to operate and manage, and better able to respond to changes in network conditions and application demands. SDN separates the control plane (e.g., routing) from the data plane (e.g., packet forwarding) in routers, centralizes the control plan logic in (typically) dedicated devices (called controllers or SDN controllers), and provides interfaces for the direct programmability of networking devices [6]. This allows for ease of network management, adaptation, and quick experimentation of new protocols and policies. SDN is rapidly being commercialized in the networking industry and is being increasingly deployed [4][5]. For instance, Google recently announced that it has completely redesigned its inter-datacenter WAN to use SDN and have reported substantial performance improvements.

In traditional IP routers/switches, data plane operations (e.g., packet forwarding) and control plane operations (e.g., routing decisions) are performed on the same device. SDN [7] decouples the control plane and the data planes similar to other prior works [7], makes the network state and intelligence logically centralized, and provides interfaces for programming networking devices. Therefore, in the SDN architecture, the data plane still resides on a network router/switch, however, control plane operations are moved to a separate SDN controller(s), which is typically a standard server. The network state (or intelligence) is logically centralized in software-based SDN controllers, which maintain a global view of the network. The network state is built by the traffic statistics sent periodically by routers/switches in the network.

PREVIOUS WORK

Moura, et.al [11] proposed Ethanol, an SDN architecture for network wide control of QoS, user mobility, AP virtualization and security. Ethanol allows QoS through integration with Hierarchical Token Bucket in OpenFlow. Virtualization is provided by having multiple virtual APs residing on a single physical device. Ethanol also tries to improve the efficiency of handoffs through a relatively rudimentary and simple approach. The controller instructs all APs, except the desired AP(s), to refuse association with the client in question and thereby forcing it to associate with only the desired AP(s). So in essence this does not make handoff any more efficient but can potentially avoid the drop in connectivity caused if a client, in a very short span of time, associates and disassociates with some intermittent APs

Suresh, et.al proposed Odin [12], an SDN framework to introduce programmability in the network. Odin's defining feature is that it proposes to improve handoff performance by allowing infrastructure to handoff clients without triggering the re-association mechanism. Odin employs the use of so-called LVAPs (Light Virtual Access Points). A LVAPs is an abstraction of a real access point that may be installed on a physical access point. When a new client enters the network the controller spawns a new LVAP with a unique SSID for this client and installs it on the first AP that the client associated with. As the client moves the controller monitors the connectivity metrics from the APs to determine to which AP the client should be handed over to next. Once determined the controller uninstalls the LVAP from the AP that the client is currently associated with and installs it on the AP that the client should associate with next. From the client's perspective it only knows one AP in the network and a handover never occurred. This eliminates the loss in connectivity experienced during the course of a traditional hand off process while the client disassociates with the current AP and tries to associate with the next AP.

Qin, et.al [13] designed a software defined approach for the IoT environment to dynamically differentiate quality levels for different IoT tasks in heterogeneous wireless networking scenarios. Since resource provisioning is NP hard a good approximation algorithm is required. They used modified network calculus to estimate transmission metrics like delay, RTT and jitter in connection with a genetic algorithm for path selection and flow scheduling to achieve efficient resource allocation and interoperability in a network with heterogeneous devices.

MOBILITY

Using Path Prediction to Improve AP handoff performance

Being in a software defined networking environment allows one the flexibility to implement finer grained improvements to attain substantial performance gains. In a conventional wireless network setup the process of handing off the client from one AP to the other is still relatively primitive. The disassociation and re-association process can take up to 5 seconds [12]. As data transfer rates are increasing, and gigabit speeds become more common the data loss that occurs during the handoff procedure is no longer negligible.

We propose that by predicting the paths that a client will take in a network we can use the programmability provided by the SDN paradigm to facilitate or even pre-empt the hand off

procedure entirely.

We reviewed two proposed SDN based solutions, Ethanol and Odin in the previous section. We felt that the efficacy of Ethanol is highly dependent of how many APs are in range of the client and sometimes sheer chance. On the other hand Odin's approach requires APs to constantly send real time connectivity metrics back to the controller and hence the volume of the data plane traffic would take up considerable bandwidth.

We think that employing something similar to the LVAP abstraction with path prediction to achieve a middle ground between these two approaches. By having the controller predict a path for the client and proactively informing the APs on the path, the data plane traffic can be considerably reduced since the controller would no longer need to monitor real time connectivity metrics to reactively initiate a hand off. The controller would have installed the LVAP on all the APs on the path and so all that would be required for the handoff to be initiated is disconnection from the current AP. Since the APs themselves are SDN switches they can be programmed to monitor connectivity metrics and disconnect the client based on a predefined threshold, which is such as to minimize data loss. This would free up bandwidth for the clients and provide them with higher throughput. Of course this method is highly dependent on a strong path prediction algorithm and a predictable environment.

We do not dismiss the approach adopted by ethanol. Even though it does not remove the data loss involved in the handoff procedure it can be used to make the overall process more efficient by associating clients with the APs on their path only. This reduces the data loss occurring when a client associates with an AP that is not on its path and hence has to disassociate soon after.

We wanted to explore both these methods to determine their efficacy in the CMUQ campus. However we did not deem it realistic to try implementing these methods within the 6 weeks that we had. Instead we decided to use data analytics and simulations (detailed in the next section) to test out our ideas.

Data Analysis

We were not able to effectively explore this avenue due certain limitations. Our initial approach was to use mobility data from the CMUQ network. We believed that there were clear paths available here that could be identified and we could work on a prediction algorithm based on them. However the CMUQ network logs proved to be inadequate for our purpose since they did not include any association/disassociation activity. Based on the correspondence with the IT staff the data consisted of polls conducted at time intervals of 10 minutes. We then tried using some public data sets from Crawdad gathered on Dartmouth University's network. We were able to identify paths in the AP logs however we could not proceed to work on the prediction due to time constraints. Moreover the Dartmouth data set did not include comprehensive tcpdump logs so actually performing an analysis of the performance of a particular solution would not have been possible. We think it would be a valuable exercise to test this approach on a much more informative data set.

Simulation

We chose Network Simulator 3 (ns3) because it was very recently updated to support the widely used C++ language instead of TCL supported by ns2. Since we did not have any experience with ns3 we had to educate ourselves on the API through the given tutorials and some degree of trial and error. In all our attempts to model a wifi network with mobile clients we realized that, as fundamental it is to modern networks, ns3 does not provide a predefined routing protocol in the stock distribution for a scenario where a client will move from one access point to the next. To articulate the issue we faced more specifically, when a client moved to another access point it sent packets destined for the host to it but the AP was unable forward it to the host. It was to our surprise that there was no example code or tutorial online dealing with this mundane situation.

We arrived at this conclusion after failing to find any such examples on any online forums or tutorials. To model such a scenario we would need to implement a routing protocol for the API. We did not deem it feasible to attempt the implementation given the time constraints and our lack of experience with NS3.

RASPBERRY PI AS AN OPENFLOW SWITCH

We attempted to put small general purpose computing devices to use as openflow enabled switches. We saw it as an embodiment of the flexibility offered by the SDN paradigm that a generic device can be repurposed to serve as a versatile network device. The low price and compact size of the R-Pi (raspberry-pi) would make it a viable hardware choice for use as a simple AP for experimental networks.

We first installed a pre-built image of openWrt from their website however it was a very basic build without even support for wireless networking and without the openFlow packages.

We narrowed down on 2 possible choices to move forward, make a custom build of openWrt with openFlow enabled* or we could use an Erlang based application, called LINC.

The latter approach, although apparently simpler, was out dated since the LINC source code had been updated to support Erlang v16A while Rasbian (a debian port for the R-Pi) only supported up to Erlang v15 B.

Pursuing the former approach we successfully built an image of the openWrt v14.07 kernel with openFlow v1.3 and support for wireless networking. The process has been posted on the project website [14] and the image has been uploaded and is available for download as well. However the R-Pi was not yet ready to serve as a wireless access point since the wireless adapters we had did not work with openWrt since the package library of openWrt did not provide drivers for the Realtek chipset that those adaptors were based on and therefore waiting on the delivery of compatible adaptors stalled progress on this front for a week after which we were able to successfully enable wireless networking on the R-Pi and start using it as a wireless access point.

When choosing the controller for our openFlow network our first choice was Floodlight since its new released promised support for openFlow v1.3 and above. However it would not successfully connect to our openFlow enabled R-Pi. Our second controller choice was Ryu. We used a prebuilt virtual machine image that was provided on the website. With Ryu we were able to successfully maintain a connection with the R-Pi. However we could not put the R-Pi to use as a

switching device by installing flow rules on it yet because that would require writing a script for the Ryu API, which due to time constraints we could not familiarize ourselves with enough. It is to be noted that when we tried testing the setup with some sample test applications we received errors, apparently originating from the R-Pi. However we have yet to ascertain if the root of these errors is some deficiency (hardware or software) in the R-Pi or somehow the application is geared towards a traditional switch and may be requesting the R-Pi to perform some tasks that it is not able to perform. Either way we feel that if we could, with a working knowledge of the Ryu API, write an application tailored to the R-Pi, we might determine the nature of the issue.

CONCLUSIONS

Using General Purpose Computing Devices as SDN switches

The main drawback of using a GPCD (General Purpose Computing Device) is that in most cases there will not be vendor support for the necessary software that needs to be installed and setup on it for it to serve as a SDN switch. There is also a lack of useful tutorials and documentation to facilitate the software setup. Therefore the setup process would, in most cases, be very tedious as it would require the user to research extensively and, as in our case, may involve trial and error to succeed in the installation and setup. On the other hand if a specialized piece of hardware, like the Linksys WRT54GL, is used the user need not worry about the issues mentioned above. Pantou is an openWRT build with openflow enabled, which is available as a prebuilt image for the WRT54GL and other common devices. Having a prebuilt image for the specific hardware significantly speeds up the setup process allowing the researcher or the user to allocate more resources to the productive purposes the network is being setup for. With the wider range of support and documentation options available customizations can be more easily made to the setup as well.

Once openWRT is installed the user interface for both the GPCD and specialized hardware is the same so neither offers any more versatility over the other. However while the UI on the specialized hardware, in most cases, is accessed over the network (using telnet or some similar service) most GPCDs can be connected directly to an external display so they are less likely to get bricked. In our experience this feature offered greater freedom to experiment and saved us considerable time that would otherwise have been spent on restoring the device after a brick.

The most significant advantage of GPCDs over specialized hardware is their processing power. The new Raspberry Pi II has an A7 chip, the same one that is used in the iPad. This could allow GPCDs to act as more versatile switches capable of supporting more complex network logic as well as allowing the functionalities of other networking middleware (such as firewalls) to be implemented on them. They may also be able to serve as “super switches” in a network allowing a certain degree of physical distribution of the control plane. This could localize control plane traffic and reduce the load on the central controller which can now be a less powerful machine.

Though they may have the computation power, GPCDs are designed as low power consumption devices. Most of the GPCDs run are powered by a USB connection which means that only so

many peripherals can be connected to their ports and their performance may not be 100%. We did an iperf benchmark on the Ethernet throughput between a Raspberry Pi and a laptop and got 43 Mbps, which is very low. Similarly if more devices are connected over the usb ports power to them is distributed and their performance decreases. The lower power consumption and having a USB power source may allow networks based on GPCDs to be highly versatile and rapidly deployable. Instead of having a power outlet for each network device, as in conventional networks, a GPCD can be coupled with a power bank in a standalone package.

Based on our experiences and research we conclude that further research into using GPCDs as SDN switches would be a worthwhile endeavor. The mobility and versatility that GPCDs provide would be of great help to researchers that are working on SDN based projects by providing them with increased flexibility. However if such flexibility is not required then we believe that a specialized device would be a better choice, especially since there is not much difference in term of the prices with a Raspberry Pi costing $30 and a WRT54GL costing $50.

Using Path Prediction and Efficient Handoffs

From the Dartmouth datasets we realized that a standalone path prediction based handoff system may not be very effective in places where there is significant overlap and intersection between paths, mostly places with open spaces where many paths intersect and originate from, while places with long corridors will benefit more. However having user profiles for network users may allow the former to overcome the aforementioned issues by making user specific predictions but that would require a large amount of resources to collect, store and analyze mobility data for each user for performance gains that may not be so significant.

We feel that the best situation to apply this method to would be outdoor wireless networks on roads. As vehicles move from AP to AP quickly extremely high rates of data loss could be expected. However by applying a LVAP based solution the intermittent data loss could potentially be avoided and hence allowing for an improved user experience.

REFERENCES

1. Open networking foundation. 2011.
2. Software defined networking. https://www.opennetworking.org/sdn-resources/sdn-definition. Last visited on: 19/4/2015.
3. Ashikur Rahman and Nael B. Abu-Ghazaleh. On the expected size of minimum-energy path-preserving topologies for wireless multi-hop networks. In 2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014, pages 889–897, 2014.
4. Jain, Sushant, et al. "B4: Experience with a globally-deployed software defined WAN." ACM SIGCOMM Computer Communication Review. Vol. 43. No. 4. ACM, 2013.
5. Hp launches comprehensive sdn portfolio. http://www.datacenterknowledge.com/archives/2012/10/03/hp-launches-comprehensive-sdn-portfolio/. Last Accessed April 20, 2015

6. Barath Raghavan, Martín Casado, Teemu Koponen, Sylvia Ratnasamy, Ali Ghodsi, and Scott Shenker. Software-defined internet architecture: Decoupling architecture from infrastructure. In Proceedings of the 11th ACM Workshop on Hot Topics in Networks, HotNets-XI, pages 43–48, New York, NY, USA, 2012. ACM.

7. Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." proceedings of the IEEE 103.1 (2015): 14-76.

8. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2):69–74, 2008.

9. Kok-Kiong Yap, Masayoshi Kobayashi, Rob Sherwood, Te-Yuan Huang, Michael Chan, Nikhil Handigol, and Nick McKeown. Openroads: Empowering research in mobile networks. SIGCOMM Comput. Commun. Rev., 40(1):125–126, January 2010.

10. Manu Bansal, Jeffrey Mehlman, Sachin Katti, and Philip Levis. Openradio: A programmable wireless dataplane. In Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12, pages 109–114, New York, NY, USA, 2012. ACM.

11. Moura, H.; Bessa, G.V.C.; Vieira, M.A.M.; Macedo, D.F., "Ethanol: Software defined networking for 802.11 Wireless Networks," *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on* , vol., no., pp.388,396, 11-15 May 2015 doi: 10.1109/INM.2015.7140315

12. Suresh, Lalith, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao. "Towards Programmable Enterprise WLANS with Odin." *Proceedings of the First Workshop on Hot Topics in Software Defined Networks - HotSDN '12* (2012). Web. 14 Aug. 2015. <http://www.engr.uconn.edu/~bing/cse330/papers/sdn/Programmable-WLAN.pdf>.

13. Zhijing Qin; Denker, G.; Giannelli, C.; Bellavista, P.; Venkatasubramanian, N., "A Software Defined Networking architecture for the Internet-of-Things," *Network Operations and Management Symposium (NOMS), 2014 IEEE* , vol., no., pp.1,9, 5-9 May 2014 doi: 10.1109/NOMS.2014.6838365

14. www.qatar.cmu.edu/mshah1