Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 1083575, 12 pages
https://doi.org/10.1155/2018/1083575

WILEY | Hindawi

Research Article

On the Tradeoff between Performance and Programmability for Software Defined WiFi Networks

Tausif Zahid,[1,2] Xiaojun Hei,[1] Wenqing Cheng,[1] Adeel Ahmad,[3] and Pasha Maruf[3]

[1]*Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*Chenab College of Engineering and Technology, Gujranwala, Pakistan*
[3]*Bahauddin Zakariya University, Multan, Pakistan*

Correspondence should be addressed to Xiaojun Hei; heixj@hust.edu.cn

Received 25 October 2017; Revised 8 December 2017; Accepted 14 January 2018; Published 29 April 2018

Academic Editor: Huimin Lu

WiFi has become one of the major network access networks due to its simple technical implementation and high-bandwidth provisioning. In this paper, we studied software defined WiFi networks (SDWN) against traditional WiFi networks to understand the potential benefits, such as the ability of SDWN to effectively hide the handover delay between access points (AP) of the adoption of the SDWN architecture on WiFi networks and identify representative application scenarios where such SDWN approach could bring additional benefits. This study delineated the performance bottlenecks such as the throughput degradation by around 50% compared with the conventional WiFi networks. In addition, our study also shed some insights into performance optimization issues. All of the performance measurements were conducted on a network testbed consisting of a single basic service set (BSS) and an extended service set (ESS) managed by a single SDN controller deployed with various laboratory settings. Our evaluation included the throughput performance under different traffic loads with different number of nodes and packet sizes for both TCP and UDP traffic flows. Handover delays were measured during the roaming phase between different APs against the traditional WiFi networks. Our results have demonstrated the tradeoff between performance and programmability of software defined APs.

1. Introduction

Many emerging Internet services have been reshaping our daily lives, which are running on wireless portable devices including mobiles and tablets. These services commonly use WiFi networks for Internet access. Studies have shown that in near future most of the IP traffic will be carried wirelessly so there exists a pushing need to increase network capacity and improve its efficiency for end-users [1]. For the streaming applications such as voice over IP, users require Internet services with quality-of-service. Providing network services to different users at the same time may cause hindrance and jitter in the traffic load. The increasing numbers of users and traffic flows everyday suggest that the traditional WiFi networks should be renovated to meet soar demands. The operational cost and traditional infrastructure of the WiFi networks have slowed down this innovation process [2], for example, the radio network resource abstraction and allocation lacks of controlling knots due to the random access on broadcast wireless medium and the dynamic channel conditions. Nowadays, there have been many intelligent devices emerging that have self-adaption awareness which also induce security challenges to the networks. In addition to deliver real-time data, the networks need to be equipped with advanced programmability features to manage these potentially hostile devices. Moreover, the network should provide measurable, manageable, and controllable interfaces to network applications at the upper layer [3].

The SDN paves a new approach of network management by partitioning the control plane and the forwarding plane. In terms of enabling programmability, ability to control network traffic and devices, SDN networks are more flexible than the traditional ones in handling constraints such as channel switching, unbalanced traffic load, and handover. The separation of the control plane and the forwarding plane not only provides flexible management but also provides the centralized control for the whole network. SDN has been accepted as a unique architecture for wired infrastructure, by

providing faster deployment of new services and applications, by enabling novel features such as virtualization. OpenFlow becomes a common south-bound protocol for the SDN deployment [4]. Software defined wireless networking is a natural extension of SDN for wireless networks, which has been proposed in networking research and industry communities in that such a separate controller can control wireless devices in a unified way.

WiFi networks have been shifting from local and independent framework to substantial public infrastructures [5, 6]. It becomes a challenge to provide services to a large-scale network with adequate coverage, low delay, and minimum disruption. A single AP can cover a radius of about 200 to 300 meters in outdoor environments, while in indoor scenarios a single AP can cover about 50 meters; hence, frequent handovers may occur due to limited sizes of hot spots. The distance between APs and end-users has a major impact on bandwidth. A user who is connected with a longer distance to its AP can only receive around 10 to 50 percent of the network bandwidth as compared to the user connected to adjacent AP. It is very important to address this persistent handover problem. In a traditional architecture, it is difficult to configure all the devices in case of small changes in network policy.

Figure 1 illustrates an SDN-based WiFi network architecture, in which multiple network management policies can be controlled under a centralized control. The devices are connected with SDN-based APs, under the management of a single controller which have a global view of networks. In a software defined WiFi network, there is no need to instrument various WiFi protocols on the APs; instead, all the packet forwarding decisions are determined by a centralized controller. By controlling the whole network in form of programmable entities, SDWN offers flexible environments for the management and performance improvement of infrastructure by deploying new services more conveniently.

In this paper, we study the tradeoff between performance and programmability for software defined WiFi networks against the traditional WiFi networks. We instrument an experimental network testbed to conduct performance comparison. This testbed contains a java-based SDN controller, APs and 24 clients for generating real network traffic. This testbed is configured with single and multiple BSS testing scenarios. The performance is measured with TCP and UDP based packets and the evaluation establishes the tradeoff between network performance and control flexibility in SDWN. We focus on a case study to examine the mobility issues in this instrumented SDWN testbed. In SDWN, there are many APs which are managed by such a central controller. This controller serves as the network brain accessible for all the APs. The SDN controller coordinates APs so that the roaming clients are able to maintain network connectivity from one AP to another. Our experiments consider two typical scenarios. In the first scenario, the same channel is used, while different channels are used in the second scenario. The comparison focus on the handover delay in two typical scenarios and the throughput is measured for a SDWN prototype, namely, Odin-V2, in comparison with the conventional WiFi network.

The rest of the paper is organized as follows. In Section 2, we provide a background review on software defined wireless networks. In Section 3, we describe the framework of an experimental SDWN network testbed. Then, we report various performance evaluation results in Section 4. Finally, we conclude the paper in Section 5.

2. Background

With rapid deployment of new Internet based applications [7–9], WiFi has become the most adopted network interface in many portable devices. WiFi services are pervasively available these days, but it is challenging to provide high-speed constant connectivity for many users. Many gaming and voice applications demand continuous network connectivity without any freezing or delay. Mobility is an essential and major issue in cell-based wireless networks. In traditional WiFi network architectures, certain handovers occur due to frequent mobility of a client when he changes his location during connection to one AP. After changing his location, if the client obtains better signal strength from another access point as compared to previously connected AP, this client may establish a new association to this new AP. In a large-scale environment of traditional WiFi networks, this client faces such a frequent handover problem which takes time due to the exchange of management frames between the client and APs. Frequent handover is one of the serious problems in traditional WiFi networks and it is not specified in the 802.11 protocol family. Handover decisions are usually made by vendor specific protocols which take decisions based on signal power, signal to noise ratio, and other criteria. SDN provides advanced network services with flexible and economical hardware, and it is based on a unified way to manage the network [10]. SDN has now been extended for WiFi and cellular networks. The extension of SDN for WLAN has been an active research because many research problems are still open to be addressed including performance and practical deployment of SDWN. OpenFlow is a typical south-bound protocol for a controller to manage network devices, while it is yet to provide the support for 802.11 protocols.

Most available SDN simulators only support wired network infrastructure. It is challenging to study the performance of wireless network because channel interference and other aspects are difficult to replicate in a simulation environment. Odin [11] is a prototype system towards a real deployment of a SDWN. Our testbed heavily utilizes the Odin architecture with necessary module upgrades and various modifications. The virtual access point (VAP) provides management competence and virtualizes the association structure of AP, which empower the administrator to program the network and deploy the WiFi infrastructure. A typical Odin infrastructure contains a single controller and multiple APs. The controller and APs communicate with each other using TCP connections. This deployment has the advantage that no modification is required on the end systems and there is no need of connection reestablishment in case of handover. The management process running on the controller will migrate the connection from one VAP to another

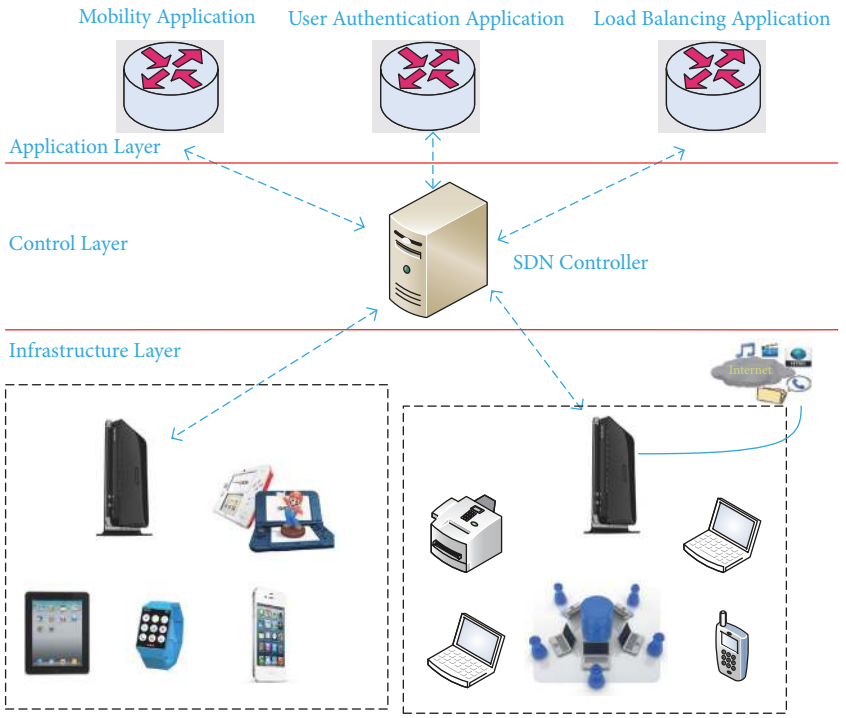


Figure 1: A software defined WiFi networking architecture.

VAP. Thereafter, this end-user can move freely within the network.

SDN provides a promising way to manage network in an elastic and cost-effective way. In recent years, the SDN has been developed beyond wired networks. We have witnessed various research efforts in implementing software defined wireless and cellular networks [12, 13]. After the separation of control and data plane customized configuration is no longer required for wireless APs. Due to this centralized nature, experiments can be run on the production network without generating traffic disturbance [14]. Network virtualization has also been examined to support wireless networks. Wireless network virtualization provides opportunities for several virtual networks to run on a shared wireless physical medium. It is promising to implement software defined wireless local area networks, while there are many challenges which need to be tackled before their widespread deployment, such as dynamically scheduling the wireless resources. With the increasing wide-range deployment and the diversification of wireless technologies, it has become a very challenging task to manage wireless networks. The software defined wireless networking poses many challenges and brings forth several research efforts to introduce the SDN features into wireless local area networks. Each access point conventionally makes decisions on its own modulation schemes, and power and channel settings based on local SNR evaluate or simply follow the default values.

The control plane of the WiFi infrastructure is much more complicated than the wired networks [12]. The first effort on instrumenting the control plane is the OpenRoads project [15], where a three-layer architecture was proposed. The flow layer implements the OpenFlow protocol to forward wireless traffic between routers. Medium specific parameters are managed using SNMP. The flow layer enables slicing traffic in order to allow an easy integration of new technologies and feasible experimentation on real networks. OpenRoads implements a similar approach like FlowVisor in wired networks [16]. The control layer centrally controls the network using the NOX controller [17] and the client can switch connections between cellular and WiFi networks to achieve seamless handovers. There are many advantages for a user to achieve enhanced coverage area and an increase in bandwidth capacity. Implementing the OpenRoads architecture requires decoupling mechanisms between service providers and network owners. This decoupling and virtualization over the laid infrastructure have far reaching effects in terms of economy and regulatory challenges faced by the industry [18]. Practical implementation of this architecture requires decoupling of service providers and network owners. OpenRadio fills this gap with the aim of providing programmability of the PHY and MAC layers by attempting to define a software abstraction layer that hides the hardware details from the upper layer programmers [19]. OpenRadio does not provide programmable PHY and MAC layers; nevertheless, it can cooperate with other projects such as WARP and CloudMAC. CloudMAC is a network architecture aimed at achieving a programmable MAC layer without resorting to software radios [20] with the introduction of the virtualized APs.

The deployment of SDWN for enterprise was studied in the Odin project [11], in which the light virtual access point (LVAP) approach was proposed, similar to LVAPs used in CloudMAC [20]. Odin and OpenRoad contribute a complete SDWN architecture. Nevertheless, there is still room for improving network delay and performance [21].

The channel-related processing may be a time critical job and the centralized processing away from the production network may significantly degrade delay sensitive applications such as VoIP or video streaming. It is not straightforward to apply the centralized control of SDN to wireless networks. Aeroflux steps up and tries to tackle the problem in a two-tier approach. AeroFlux was built based on the Odin framework [11]. This architecture divides the control plane in two layers. The lower layer, handled by nearsighted controllers (NSCs), is liable for situations that do not require global state data or those events that occur very frequently [22]. The services like load balancing and network monitoring, which are controlled by a central authority, are controlled by the global controller.

CloudMAC [23] is another software defined wireless network prototype in which APs are responsible for forwarding MAC frames. In this architecture, MAC frames are processed on the servers in data centers. CloudMAC WTPs require a WLAN driver and a small application for controlling infrastructure, which reduces software bugs and software complexity. Behop [24] is another SDWN architecture used for a wide set of management modules of channel, power, and association control in different environments. Utilizing the VAP abstraction to decouple WiFi logic from the physical infrastructure and control, the infrastructure is exposed to users. Behop also runs alongside production networks. Behop APs serve as OpenFlow switches and extend SDN functionalities to expose primitives for the channel, power, and association control.

In [25–27], authors proposed different SDWN architectures which utilize OpenWrt based embedded systems. In [25], Lee et al. developed an access point using Raspberry Pi. In [26] the instrumented SDWN platform can control channel assignment and interference management. In [27], Sundaresan et al. implemented and evaluated the performance of wireless home routers, in which the results showed how the characteristics of home wireless networks affect the performance of user traffic in real home environments. In [28, 29], the authors also utilized OpenWrt based systems and both architectures slice their network bandwidth in a software defined approach.

Our study includes various experiments to evaluate and compare the performance of our testbed with the existing WiFi infrastructures. The testbed is equipped with the latest OpenWrt firmware, packages, modifications in the specified modules, and drivers. Our testbed does not require any client-side modifications and our approach also removes the handoff delay with different channels in multi-BSS scenarios [30]. Previous studies considered different approaches towards the SDWN architectures with different scopes. Our testbed specifically focused on the handover performance of software defined WiFi networks with different parameters (such as number of clients, VAP, and packet size) in order to push the loading stress on single and multi-BSS scenarios [30–32].

3. Testbed

In order to enable programmability in the WiFi infrastructure, we construct our testbed without any client-side modifications. The network performance is evaluated with different workloads and types of traffic on the testbed. Distance and interference are the factors which have major impact on the throughput. The testbed includes updated versions of the Odin architecture with upgraded control functionalities, different applications running on the controller, the upgraded OpenWrt system, and the utility modules.

As shown in Figure 2, our testbed consists of three major parts. The first part is an SDN-based controller which centrally controls the whole network through different policy based applications. The second part includes a number of commodity access points, in which the NetGear routers (WNDR3700v4) serve as OpenFlow switches by instrumenting an OpenWrt based operating system. The OpenWrt release 15.05 is used for implementing the OpenWrt based image for the embedded Linux system. Different utilities are installed in order to make the devices function as OpenFlow switches. Major open source projects used in our testbed include the OpenVswitch version 2.3, the ath9k Linux driver, and the user-level click modular router. The NetGear WNDR 3700v4 model is equipped with the Atheros AR8327 chipset, 560 MHz CPU and 128 Mbit RAM. TP-Link TL-SG1024DT switches are used in order to provide the SSH utility and the Internet access to end-users. The third part includes 8 mini PCs which install the Intel Core i3 processor with 4 GB of RAM and 18 wireless USB adapters are used to serve as WiFi end-hosts. In order to send traffic from the iPerf clients to the iPerf server in a controlled manner, we utilized the clusterssh utility. Clusterssh provides the utility to issue the same command into several end-hosts in parallel. Otherwise, we have to log in each end-host with SSH and configure these hosts serially from a single input window over a SSH connection. Our experiments only test the 802.11 network at the 2.4 GHz range.

We also set up a multiple BSS network topology to study the handover performance when WiFi clients roam between APs as shown in Figure 3, in which there are two APs which are controlled by a SDN controller. Initially, a client is connected to AP-1. After some time this client moves its location towards AP-2. A laptop is used here for the handover experiments. In this study, dynamic IPs are assigned to the laptop by the TP-Link router. For displaying measurement results and maintaining network connectivity, we instrumented scripts to send ICMP messages periodically from one node to another.

4. Results

We conduct various measurement sessions and report the results of our SDWN testbed in this section. Our experiments were conducted with the following operational settings. Our testbed consists of multiple clients which are connected to the network, a master node which is responsible for the LVAP assignment and the installation of OpenFlow forwarding rules, and different applications which run on the master node. In order to send traffic at a time from different iPerf clients to the iPerf server, we utilize the clusterssh utility. Clusterssh provides the console to issue the same commands into several machines in a batch; otherwise, we have to log in each node with a standard SSH utility and issue the

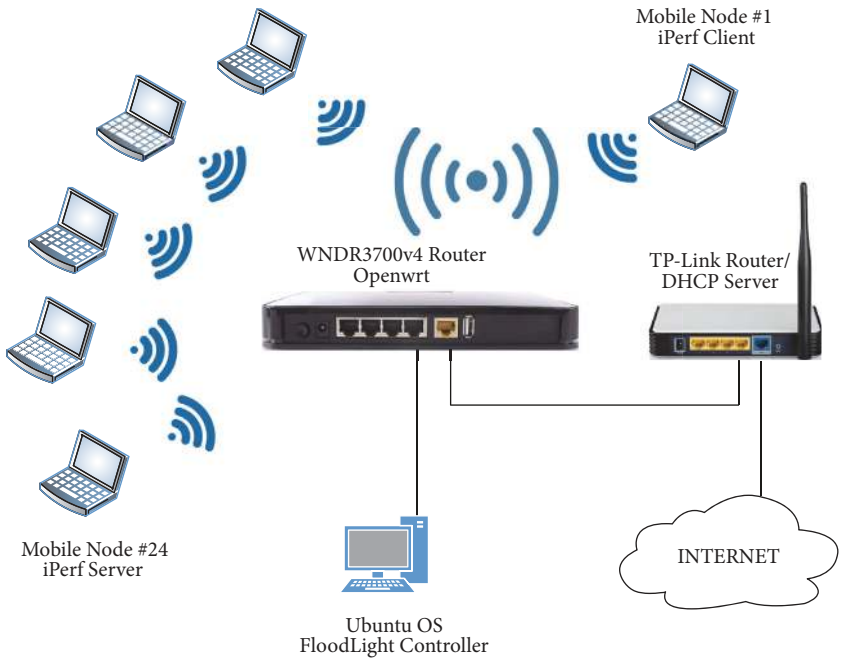


Figure 2: A single BSS network topology.

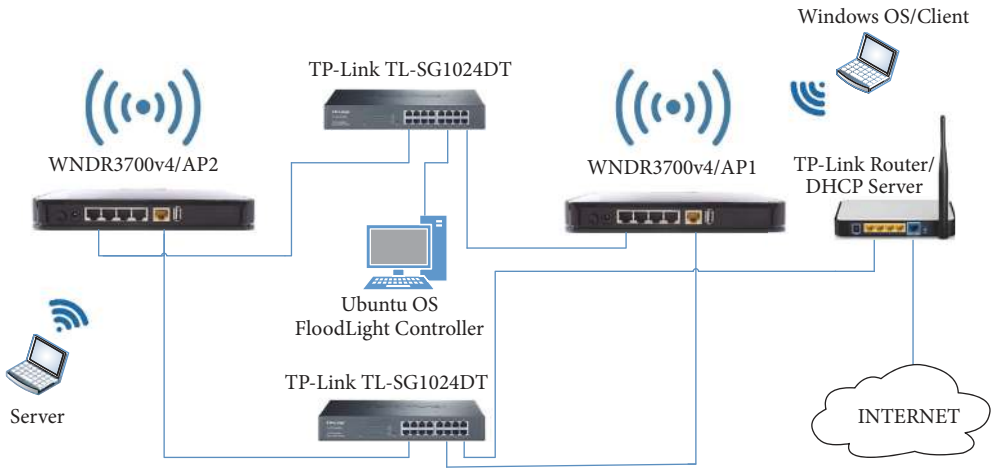


Figure 3: A multi-BSS network topology.

commands serially from a single input window over an SSH connection.

This testbed not only has instrumented the programmable WiFi nodes towards a software defined WiFi network, but also includes different techniques to improve the network performance. These experiments are designed to provide some insight into understanding the practicality of software defined systems in different aspects. In order to examine the design space of the software defined systems, we also instrumented a traditional WiFi network as the benchmark for the comparison purpose. Every access point has its own architecture or mechanisms; hence, its performance varies for association and reassociation with the clients. Without explicit statements, we use the same lab settings with different architectures in order to achieve a fair comparison. In this measurement study, we repeat our experiment sessions 10 times for each combination of the parameters, the reported values are the computed as the average of these 10 experiments, and MATLAB is used for plotting of result figures. We built this testbed with a simple NetGear WNDR 3700v4 switch. First, we measured the performance with the commodity hardware. Then, the OpenWrt based image was installed in the same NetGear switches for comparison with the Odin-V2 architecture. This Odin-V2 architecture is built based on OpenWrt. We aim to study whether the performance degradation was due to

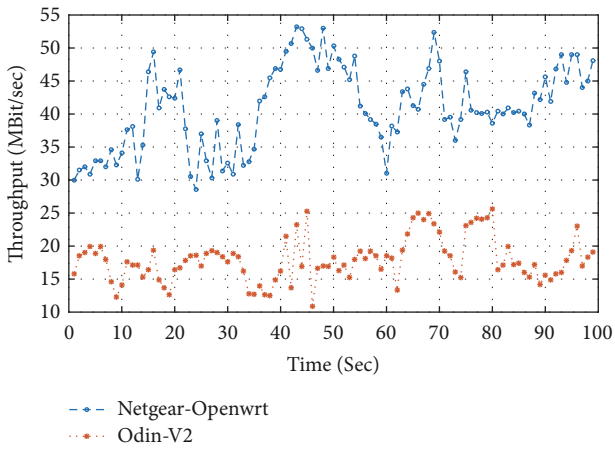


FIGURE 4: Network throughput with packet size $L = 1500$ bytes: NetGear-OpenWrt versus Odin-V2.

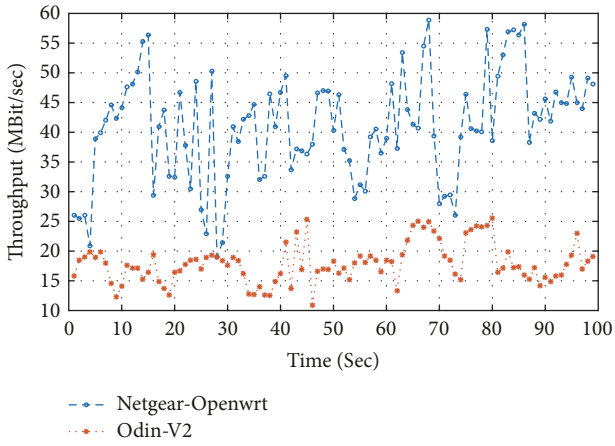


FIGURE 5: Network throughput with packet size $L = 1500$ bytes: generic NetGear versus Odin-V2.

OpenWrt or due to software implementation. Our evaluation starts with the throughput performance with different packet sizes and different topology settings.

Figure 2 shows the topology of our network testbed with a single BSS. In the first experiment, we set up a single access point with 2 clients, in which one is serving as the client and the other serves as the server. These two clients are associated with the same AP. One client is generating the packet streams and the other client receives the packets. The performance evaluation was conducted for 100 seconds with the packet size 1500 bytes in each session.

As shown in Figures 4 and 5, the throughput performance of Odin-V2 with the commodity NetGear WNDR 3700v4 switch and the NetGear WNDR 3700v4 with the installed OpenWrt firmware. The generic NetGear and the OpenWrt firmware were used to examine the delay performance benchmark to evaluate the software defined testbed based on OpenWrt. We aim to investigate that the performance degradation of software defined approach is either due to OpenWrt or some other reasons. The generated traffic is the overall network traffic because there are only two clients in

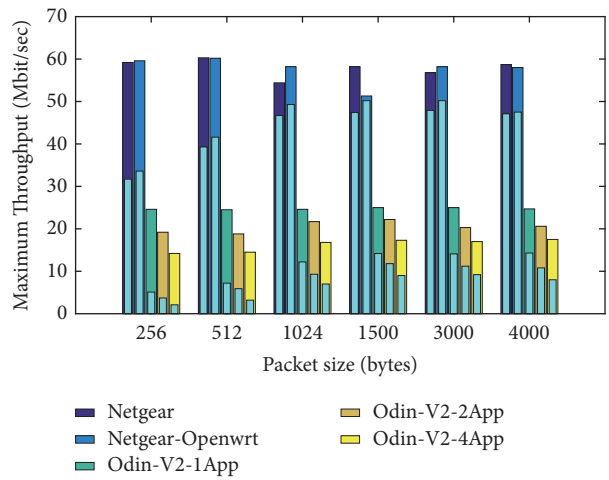


FIGURE 6: Maximum network throughput with 1 client: UDP versus TCP. Outer bar (blue bar) depicts the throughput for UDP flows and the inner bar shows the throughput of TCP flows.

this network. Afterwards, we increase the number of clients in the same topology. Among these clients, one client becomes the receiver or the iPerf server and the other remaining clients will generate traffic or behave as iPerf clients. All the traffic will be sent simultaneously towards the iPerf server by using the clustersh utility. In our results, outer bars depict the throughput for UDP flows and the inner bar shows the throughput of TCP flows.

Figure 6 illustrates the maximum network throughput of 1 client with UDP and TCP flows. This figure also contains the comparison of conventional network and Odin-V2 which includes 4 applications running on the controller. These applications create virtual access points. The applications running atop master include load balancing, authentication, and mobility. We also investigated the impact on network throughput by generating multiple slices on a single router. This kind of virtual access points can run on single or multiple access points within the same network depending on user's desire or need. The maximum throughout for the traditional network reaches around 60 Mbps; nevertheless, the performance of software defined approach is almost half of the conventional network. This performance drop is due to controlled traffic in SDN because multiple LVAPs are created. Other reasons may be the software implementation of switching in AP. The software defined approach running with multiple applications has less throughput as compared to single application running on controller. For each individual client, an individual virtual AP is created to deploy specific set of packet process rules. Figures 7 and 8 illustrate the performance of the 6 and 12 clients for UDP and TCP flows with respect to packet sizes. In these scenarios, we increased the traffic generating clients from 6 to 12. The measurement results show that Odin-V2 network is approaching 15 Mbps for smaller packet sizes and the average network throughput is more than half for cases with smaller packet sizes. In case of larger packet sizes, the throughput difference between UDP and TCP flows drops down to half. The relationship between

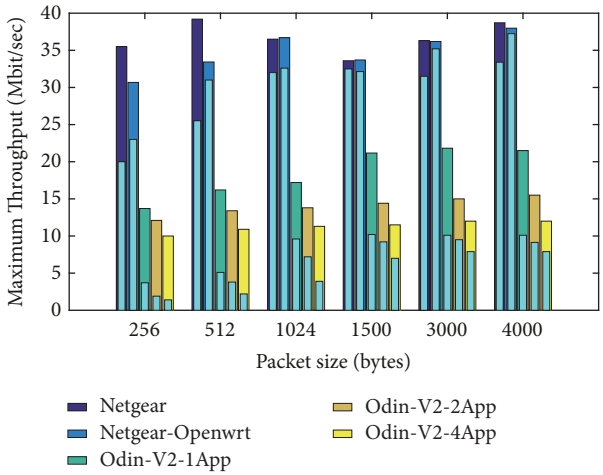


Figure 7: Maximum network throughput with 6 clients: UDP versus TCP. Outer bar (blue bar) depicts the throughput for UDP flows and the inner bar shows the throughput of TCP flows.

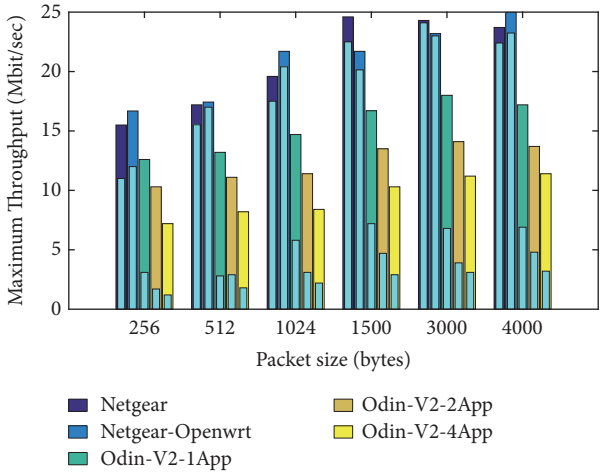


Figure 8: Maximum network throughput with 12 clients: UDP versus TCP. Outer bar (blue bar) depicts the throughput for UDP flows and the inner bar shows the throughput of TCP flows.

throughputs versus packet size is proportional. However, when the packet size is too large, it becomes difficult to deliver the packets to the receiver side. In case of noisy channel conditions and large packet sizes, the throughput reduces more than previous conditions due to increasing retransmission. Usually one AP can provide services to about 25 clients at a time but it also depends on the AP architecture. So in order to test the network performance under higher stress conditions, we utilize around 24 clients. Figure 9 also illustrates the performance evaluation of 24 clients with both architectures. We observed that Odin-V2 with different applications is still working. The TCP throughput is around 5 Mbps and the UDP throughput is less than 10 Mbps with different packet sizes but even with this throughput many multimedia applications can work well within this range.

Generally speaking, all the devices in a WiFi network do not remain in the active mode simultaneously. These devices

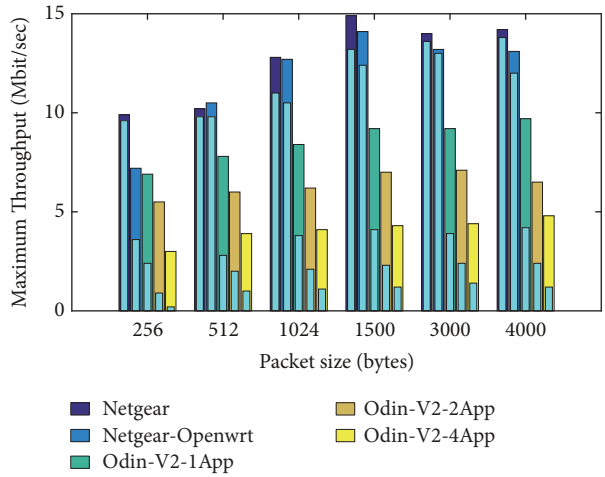


Figure 9: Maximum network throughput with 24 clients: UDP versus TCP. Outer bar (blue bar) depicts the throughput for UDP flows and the inner bar shows the throughput of TCP flows.

switch as active and passive clients from time to time within the network. In Figures 10 and 11, we aim to study the impact of passive clients on the network throughput. Hence, we increased the number of clients from 6 to 24 clients by having half of the clients in the passive mode. These results show the falling trend of throughput of about 6 to 10 Mbps in case of passive clients. Figure 12 also illustrates the same performance degradation trend for different types of flows with constant 1500 bytes packet size. The primary drop in performance of the conventional WiFi network can be observed during the increment in the number of clients from 1 to 6.

Figure 13 shows the throughput comparison of all active clients with different combination of active and passive clients. The inner bar depicts the performance of all active clients and the outer bar shows the performance of combination of active and passive clients. These experiments demonstrate the impact of the increasing number of passive clients on the throughput of active clients. These results also show that the conventional network has more impact on the performance degradation as compared to the software defined approach in case of a small number of clients.

Figure 14 shows the impact on the throughput and the reassociation delay during handover. Generally handover occurs due to signal strength. When a client receives better signal strength from an AP, as compared to previously associated AP, it reassociates itself towards another AP. This handover mechanism is not specified in the 802.11 protocols. It depends on the client architecture how it behaves under this specific situation. There are two major factors behind the variation of handover delay: one is the delay occurrence due to AP and the other is due to initialization of client. In our study, a client initiates this procedure. In order to minimize the effect of different architectures, we have utilized the same hardware for all cases. In this experiment, the client moves from one AP towards another after approximately 25 seconds. The results show that the software defined approach has negligible fluctuation during handover. One reason behind

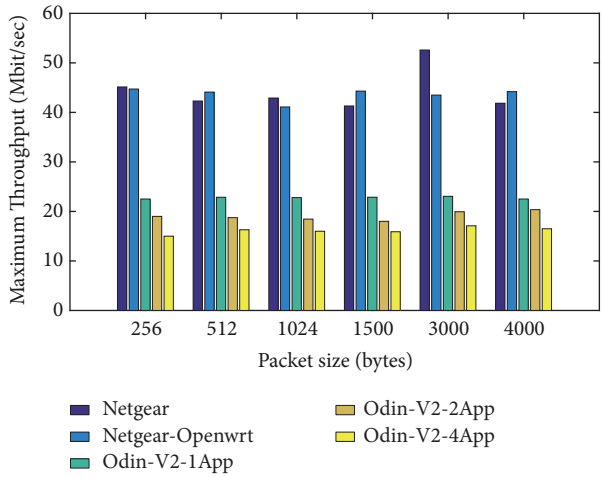


Figure 10: Maximum network throughput with 3 active and 3 passive clients.

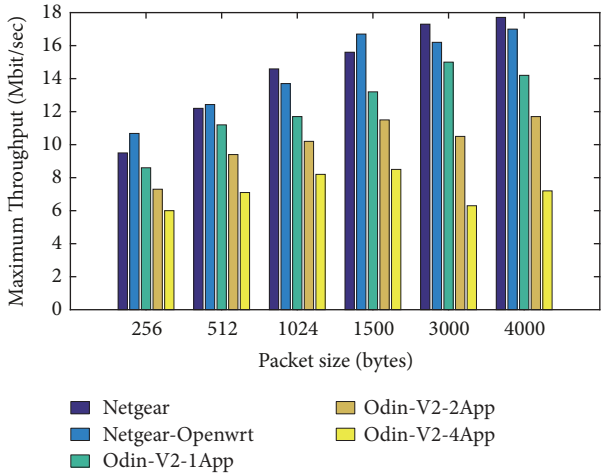


Figure 11: Maximum network throughput with 12 active and 12 passive clients.

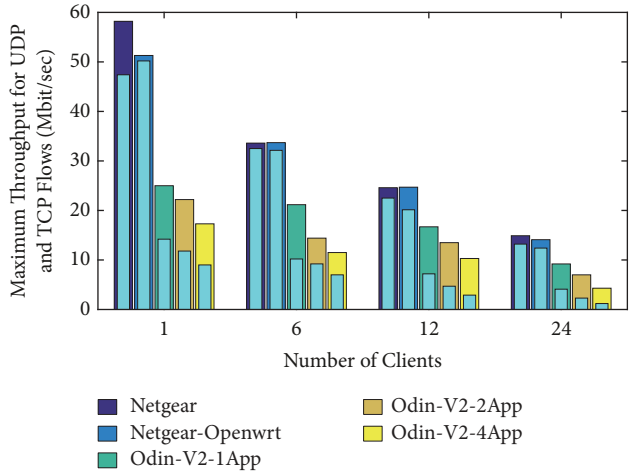


Figure 12: Network throughput with packet size $L = 1500$ bytes. Outer bar (blue bar) depicts the throughput for UDP flows and the inner bar shows the throughput of TCP flows.

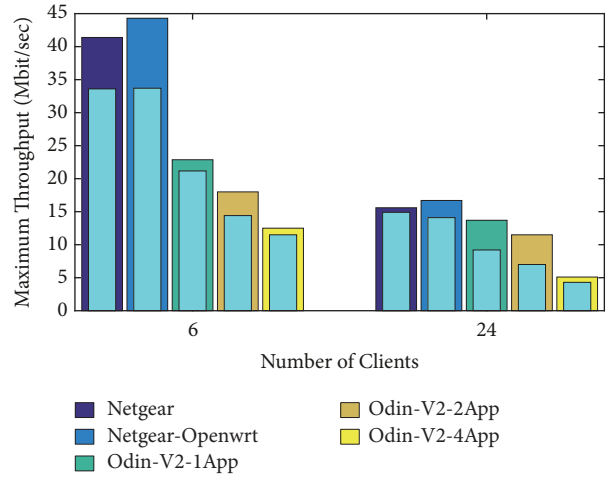


Figure 13: Network throughput with different number of active and passive clients. The inner bar depicts the performance of all active clients and the outer bar shows the performance of combination of active and passive clients.

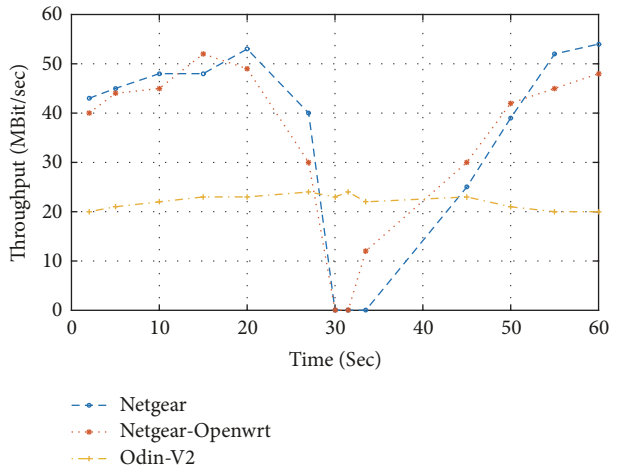


Figure 14: Network throughput during the handover.

this normal transition state is that there is no exchanging of layer-2 and layer-3 messages. The other reason is due to the centralized control, whereas the conventional network takes around 2 to 3 seconds for reassociation and takes more time to regain the prehandover throughput. Although Odin-V2 provides less throughput as compared to the conventional network, this handover delay provides a practical solution for many streaming applications, where minor delay can affect the performance of applications, such as VoIP and online gaming. Figure 15 shows the round-trip time, where the purpose of this experiment is to find the end-to-end latency, because it has a negative effect on throughput. For small packet sizes, the end-to-end latency is almost the same but for large packet sizes Odin-V2 show 4 times more latency as compared to conventional network. Figure 16 shows the handover delay; in this case the two access points have different channels. As mentioned earlier, Odin-V2 does not exchange layer-2 and layer-3 messages between clients

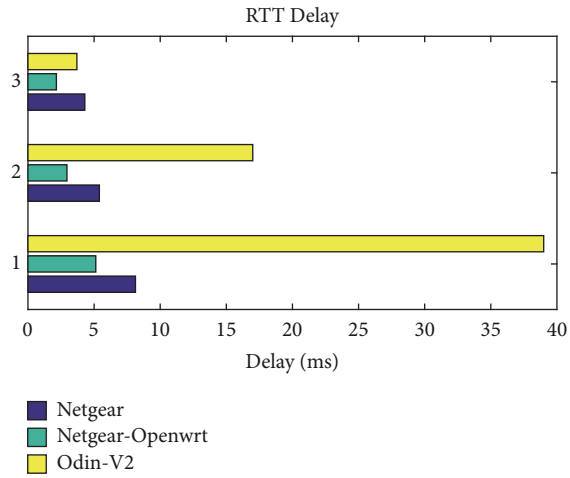


Figure 15: The round-trip time performance comparison.

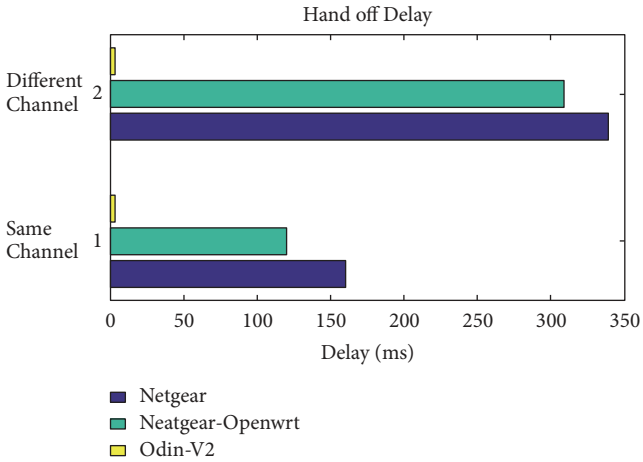


Figure 16: Handover delay on the same operation channel and the different operation channels.

and APs, and the handover delay is practically negligible compared with the conventional WiFi network. However, the small delay as shown in Figure 16 is due to the communication between the AP and the controller, which is responsible for association/dissociation of a client from one AP to another.

With lacking of coordination between APs, a traditional WiFi network can experience unbalance traffic load. Figures 17 and 18 illustrate two scenarios, before and after load balancing with different number of access points. These figures depict a hypothetical scenario for balancing network load among different access points, showing how SDN helps to balance its load in order to improve the network throughput among different clients. This hypothetical scenario is tabulated in Table 1, showing the scenario of three access points, and Table 2 shows the scenario of 4 access points.

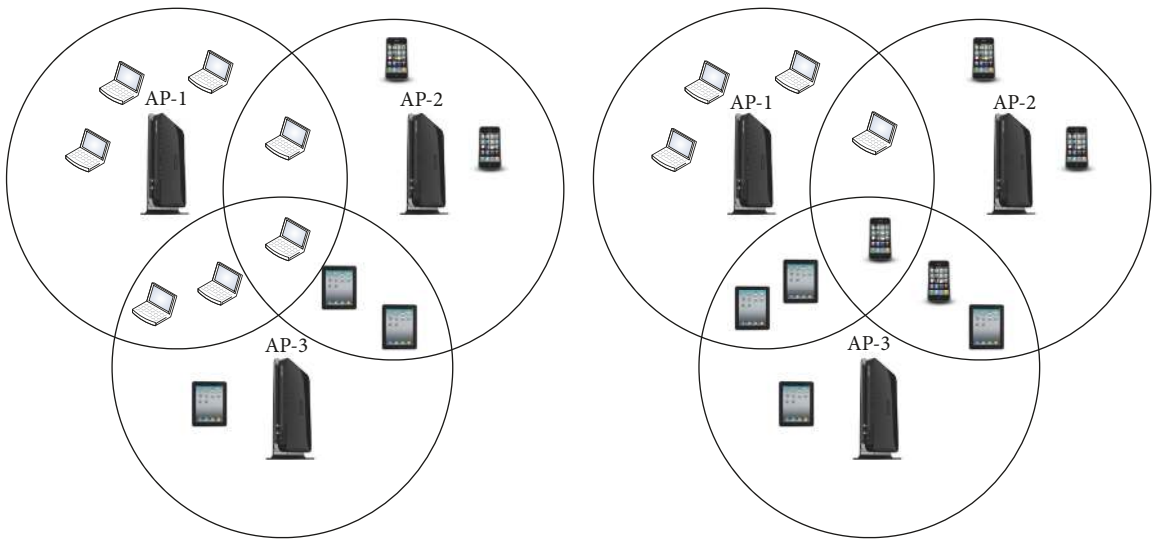
In summary, Figure 19 shows the tradeoff between programmability and performance. Odin-V2 shows lower throughput as compared to the traditional WiFi network devices but does also show the flexibility of a programmable device. The throughput difference between the NetGear-OpenWrt implementation and the SDN-based wireless network is due to the OpenVswitch and the click modular router because the click modular router is running at the user space of Linux. The degradation on the throughput performance by SDN is tolerable because this VAP implementation provides a smooth handover, which indicates the user connection migration from one AP to another without any noticeable delay.

5. Conclusion

In this paper, we conducted a measurement study of an SDWN testbed with different packet sizes, virtual access points, and number of clients to investigate how different SDN-based WiFi networks behave with different traffic loads with typical network settings. We investigated the handover mechanism of SDWN and traditional WiFi networks by the deployment of multiple access points. Our practical deployment experiences depicted the behaviors of our SDWN infrastructure with several applications running on the controller. We also observed the performance implication of deploying different applications on the controller in SDWN. The average and maximum throughput performance of TCP and UDP flows was evaluated for different network testing scenarios. Our testbed platform, Odin-V2, is based on OpenWrt; hence, the performance of Odin-V2 was therefore gauged using OpenWrt as the benchmark.

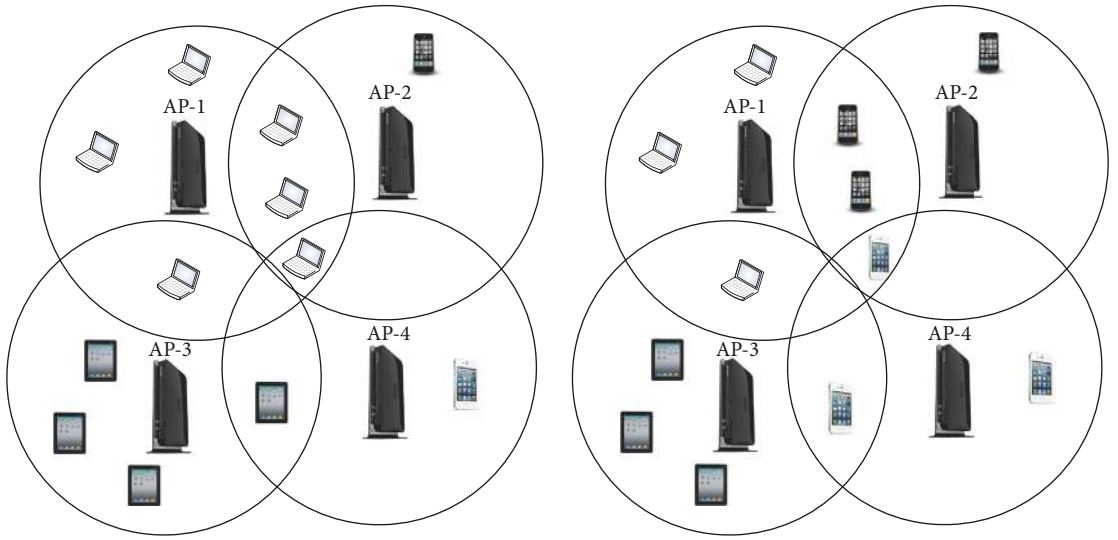
The logically centralized nature of the SDWN provides many benefits for management at the cost of performance degradation. In particular, the performance with a large number of clients is still an open issue. SDWN may be a promising approach to solve many issues including handover, load balancing, and managing complexity. However, before large-scale deployment of this software defined approach, several issues including throughput, delay, resource discovery, and security need to be addressed. Latency in SDWN is still an open research challenge for many applications which is also demonstrated by our study. The SDWN architecture provides a rich set of control features, while traditional WiFi networks are still advantageous in better network performance. This comparison study of different SDWN implementations shows that no individual architecture can fulfill the demand of users, so network administrator should devise application specific architectures for network optimization. We conjecture that the performance degradation of this software defined approach is due to the click modular router used in our testbed. Such a software-based implementation in the user space of Linux is a major issue; system overhead may also arise due to generation of different LVAPs created for each user.

In order to gain full advantages of the SDWN architectures, specific routers can be designated for specific applications. This would increase the share of deterministic component of the overall load compared to stochastic nature for the rest of the load. As deterministic models are easier to predict, it makes them more reliable compared to their stochastic counterparts. The programmable nature of the proposed SDWN architecture also enables us to counter for the inherent drawbacks of a deterministic model. This can be achieved by scheduling certain applications to start on certain



(a) Before (b) After

Figure 17: Load balancing with 3 access points.



(a) Before (b) After

Figure 18: Load balancing with 4 access points.

Table 1: Load balancing case I.

Device	Without load balancing			With load balancing		
	AP1	AP2	AP3	AP1	AP2	AP3
Generic NetGear	8.32 Mbps	19.43 Mbps	29.11 Mbps	8.31 Mbps	19.42 Mbps	29.13 Mbps
NetGear-OpenWrt	7.31 Mbps	8.31 Mbps	25.6 Mbps	7.31 Mbps	8.32 Mbps	25.61 Mbps
Odin-V2	3.52 Mbps	8.33 Mbps	12.52 Mbps	6.25 Mbps	6.25 Mbps	6.25 Mbps

Table 2: Load balancing case II.

Device	Without load balancing				With load balancing			
	AP1	AP2	AP3	AP4	AP1	AP2	AP3	AP4
Generic NetGear	9.7 Mbps	14.55 Mbps	58.2 Mbps	58.2 Mbps	9.7 Mbps	14.55 Mbps	58.2 Mbps	58.2 Mbps
NetGear-OpenWrt	8.5 Mbps	12.82 Mbps	51.3 Mbps	51.3 Mbps	8.5 Mbps	12.82 Mbps	51.3 Mbps	51.3 Mbps
Odin-V2	4.16 Mbps	6.25 Mbps	25 Mbps	25 Mbps	8.33 Mbps	8.33 Mbps	8.33 Mbps	8.33 Mbps

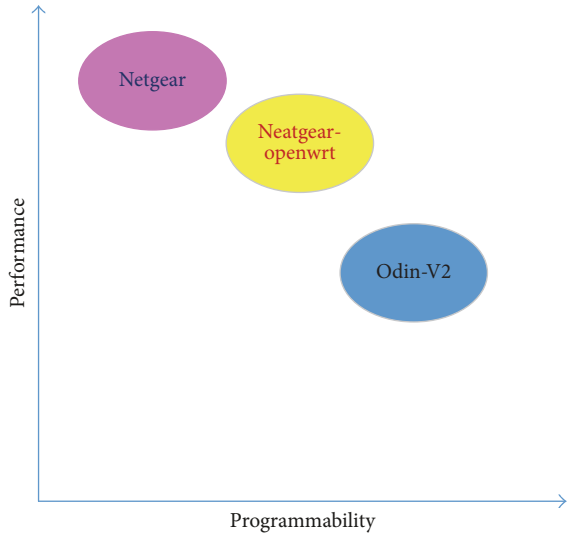


Figure 19: Performance versus programmability.

routers based on variety of parameters including users and time. The network load can be further balanced and thus the network performance may be improved by introducing hybrid traffic balancing of passive and active modes. The throughput performance of the proposed network model can also be further improved using better network management policies. We plan to design and implement intelligent load balancing schemes on Zynq-based programmable WiFi systems in a software/hardware codesign approach [33].

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (no. 61370231).

References

[1] "Cisco Visual Networking Index: Forecast and Methodology," 2017, https://www.cisco.com.

[2] N. Feamster, J. Rexford, and E. Zegura, "An intellectual history of programmable networks," *Queue*, vol. 11, no. 12, Article ID 2560327, 2013.

[3] T. Zahid, F. Y. Dar, X. Hei, and W. Cheng, "An empirical study of the design space of smart home routers," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 9677, pp. 109–120, 2016.

[4] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using open flow: a survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 493–512, 2014.

[5] Y. Gao, L. Dai, and X. Hei, "Throughput Optimization of Multi-BSS IEEE 802.11 Networks with Universal Frequency Reuse," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3399–3414, 2017.

[6] Y. Zhang, "GroRec: a group-centric intelligent recommender system integrating social, mobile and big data technologies," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 786–795, 2016.

[7] Y. Zhang, M. Qiu, C.-W. Tsai, M. M. Hassan, and A. Alamri, "Health-CPS: Healthcare cyber-physical system assisted by cloud and big data," *IEEE Systems Journal*, vol. PP, no. 99, 2015.

[8] M. U. Aslam, A. Derhab, K. Saleem et al., "A Survey of Authentication Schemes in Telecare Medicine Information Systems," *Journal of Medical Systems*, vol. 41, no. 1, article no. 14, 2017.

[9] C. Xu, Q. Chen, H. Hu, J. Xu, and X. Hei, "Authenticating Aggregate Queries over Set-Valued Data with Confidentiality," *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[10] S. Sezer, S. Scott-Hayward, P. Chouhan et al., "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, 2013.

[11] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise WLANS with Odin," in *Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks (HotSDN '12)*, pp. 115–120, ACM, Helsinki, Finland, August 2012.

[12] C. J. Bernardos, A. de la Oliva, P. Serrano et al., "An architecture for software defined wireless networking," *IEEE Wireless Communications Magazine*, vol. 21, no. 3, pp. 56–61, 2014.

[13] Y. Zhang, M. Chen, N. Guizani, D. Wu, and V. C. Leung, "SOVCAN: Safety-Oriented Vehicular Controller Area Network," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 94–99, 2017.

[14] Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. McKeown, "Slicing home networks," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Home Networks (HomeNets '11)*, pp. 1–6, August 2011.

[15] K.-K. Yap, M. Kobayashi, R. Sherwood et al., "OpenRoads: Empowering research in mobile networks," *SIGCOMM Computer and Communications*, vol. 40, no. 1, pp. 125-126, 2010.

[16] R. Sherwood, M. Chan, and A. Covington, "Carving research slices out of your production networks with openflow," *Computer Communication Review*, vol. 40, no. 1, pp. 129-130, 2010.

[17] N. Gude, T. Koponen, and J. Pettit, "NOX: towards an operating system for networks," *Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[18] C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, C. N. A. Corrêa, S. Cunha De Lucena, and R. Raszuk, "Revisiting routing control platforms with the eyes and muscles of software-defined networking," in *Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks, HotSDN 2012*, pp. 13–18, Finland, August 2012.

[19] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "OpenRadio: a programmable wireless dataplane," in *Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks (HotSDN '12)*, pp. 109–114, Helsinki, Finland, August 2012.

[20] P. Dely, J. Vestin, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo, "CloudMAC - An OpenFlow based architecture for 802.11 MAC layer processing in the cloud," in *Proceedings of the 2012 IEEE Globecom Workshops, GC Wkshps 2012*, pp. 186–191, USA, December 2012.

[21] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the 1st ACM International*

Workshop on Hot Topics in Software Defined Networks, HotSDN 2012, pp. 7–12, Finland, August 2012.

[22] J. Schulz-Zander, C. Mayer, B. Ciobotaru, S. Schmid, and A. Feldmann, "OpenSDWN: programmatic control over home and enterprise WiFi," in *Proceedings of the the 1st ACM SIGCOMM Symposium on Software Defined Networking Research (SOSR '15)*, pp. 1–12, Santa Clara, Calif, USA, June 2015.

[23] J. Vestin, P. Dely, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo, "CloudMAC," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 4, p. 42, 2013.

[24] Y. Yiakoumis, M. Bansal, A. Covington, J. Van Reijendam, S. Katti, and N. McKeown, "BeHop: A testbed for dense WiFi networks," in *Proceedings of the 9th ACM MobiCom Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization, WiNTECH 2014*, pp. 1–8, USA, September 2014.

[25] W. J. Lee, J. W. Shin, H. Y. Lee, and M. Y. Chung, "Testbed implementation for routing WLAN traffic in software defined wireless mesh network," in *Proceedings of the 8th International Conference on Ubiquitous and Future Networks, ICUFN 2016*, pp. 1052–1055, Austria, July 2016.

[26] R. Riggio, M. K. Marina, and T. Rasheed, "Interference management in software-defined mobile networks," in *Proceedings of the 14th IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, pp. 626–632, Canada, May 2015.

[27] S. Sundaresan, N. Feamster, and R. Teixeira, "Measuring the performance of user traffic in home wireless networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 8995, pp. 305–317, 2015.

[28] K. L. Huang, C. L. Liu, C. H. Gan, M. L. Wang, and C. T. Huang, "SDN-based wireless bandwidth slicing," in *Proceedings of the International Conference on Software Intelligence Technologies and Applications & International Conference on Frontiers of Internet of Things*, pp. 77–81, Hsinchu, Taiwan, 2014.

[29] H. H. Gharakheili, L. Exton, and V. Sivaraman, "Managing home routers from the cloud using Software Defined Networking," in *Proceedings of the 13th IEEE Annual Consumer Communications and Networking Conference, CCNC 2016*, pp. 262-263, USA, January 2016.

[30] T. Zahid, X. Hei, and W. Cheng, "Understanding performance bottlenecks of a multi-BSS software defined WiFi network testbed," in *Proceedings of the 1st IEEE International Conference on Computer Communication and the Internet, ICCCI 2016*, pp. 153–156, China, October 2016.

[31] T. Zahid, F. Y. Dar, X. Hei, and W. Cheng, "A measurement study of a single-BSS software defined WiFi testbed," in *Proceedings of the 1st IEEE International Conference on Computer Communication and the Internet, ICCCI 2016*, pp. 144–147, China, October 2016.

[32] T. Zahid, X. Hei, and W. Cheng, "Understanding the Design Space of a Software Defined WiFi Network Testbed," in *Proceedings of the 14th International Conference on Frontiers of Information Technology, FIT 2016*, pp. 170–175, Pakistan, December 2016.

[33] J. Kang, X. Hei, and J. Song, "A Comparative Study of Zynq-Based OpenFlow Switches in a Software/Hardware Co-design," in *International Workshop on Network Optimization and Performance Evaluation (NOPE)*, vol. 10658 of *Lecture Notes in Computer Science*, pp. 369–378, Springer International Publishing, 2017.