A Survey of Software-Defined Wireless Networks

Adam Drescher, adrescher (at) wustl.edu (A paper written under the guidance of Prof. Raj Jain)

Download 

Abstract

Software-Defined Networking (SDN) has been applied to many different areas of wireless networking, bringing them greater flexibility, fine-grained control, and unparalleled ease-of-use. However, SDN incurs some overhead, so its potential benefits must be evaluated in each networking setting. Fortunately, the benefits of SDN appear to outweigh the overheads incurred in all wireless network environments tested so far. In infrastructure-dense wireless networks, SDN can provide per-flow routing while reducing the distance between the OpenFlow switches and the controllers; this reduces customer latency while providing intuitive controls to the company. In infrastructure-less environments, SDN can provide natural virtualization and traffic isolation, providing security and configurability in networks that are not cooperative with such goals. In all wireless environments, SDN helps meet various traffic engineering goals, e.g. load balancing, with ease. In this paper, we survey applications of SDN to a variety of wireless networks, and note the benefits of each approach.

Keywords

Software-Defined Networking, Wireless Networks, OpenFlow, Slicing, Wireless Mesh Networks, Wireless Local Area Networks, Wireless Personal Area Networks, Cellular Networks, Campus Networks, Rural Wireless Network Operators, Traffic Engineering

Table of Contents

1 Introduction

Wireless networks are particularly challenging. They must cope with more dynamicity than wired networks, particularly due to the unreliable nature of the wireless medium. In general, more dynamicity means more control traffic. When different wireless networks are not properly coordinated, they can interfere with each other's signals, degrading the wireless experience for both hosts. Yet at the same time, wireless networks have become so ubiquitous, they have a wide user base that wants high speeds for consuming increasingly-rich web content. This problem is industry wide. Cellular networks are tending towards smaller cells to find faster data rates, but smaller cells increase interference [Kolias13]. The typical user of a Wireless Local Area Network is a non-expert, so the ubiquity of these wireless networks has further increased chances for signal interference. Even ignoring interference with the signal, this consistent push for more speed often decreases the transmission range of the signal [Dely13]. Similarly, more wireless networks are becoming oversaturated with devices. As a result, network operators are tending towards larger channel sizes to increase network capacity. However, this can also lower the transmission range. As these transmission ranges shorten, networks need a dense deployment of access points (APs) to cope with these problems, leading to more expenses. Wireless networks with dense deployments of access points are called infrastructure-heavy wireless networks. Intuitively, wireless networks that are light on infrastructure would need other strategies to provide connectivity to users.

Even with such variability between these wireless networks, nearly all of them must deal with poor vendor interoperability. Networking protocols and tools are too complex to be properly extended, so network operators must rely on the tools that vendors provide. Similarly, use of the vendor tools is at many times heuristic. As a result, it is difficult for operators to share experiences with each other: it is unlikely the solution applies to the other operator because they will have different tools and hardware. Vendor lock-in, i.e., being very reliant on a single vendor for equipment and tools, is another consequence of poor vendor interoperability. Companies should be able to buy whatever equipment makes sense and not worry about if the equipment can work together. Software-Defined Networking is an abstraction that can eliminate many of these woes. It can provide flexibility, better management capabilities, and ease-of-use to all of these wireless networking environments. It can also help balance wireless signaling concerns and data rate concerns for many networks; the largest difference of these concerns is between infrastructure-heavy and infrastructure-less wireless networks.

In this paper, we will survey the current applications of Software-Defined Networking to wireless networks. Section 2 offers a primer on Software-Defined Networking and addresses some common misconceptions. Readers familiar with SDN are still encouraged to read the Common Misconceptions subsection. In Section 3, we will discuss the broad, over-arching SDN techniques that are applied to wireless networks. These techniques are frequently employed by the research in later sections. Section 4 surveys current applications of SDN to wireless networks with dense infrastructure. In Section 5, we address applications of SDN to wireless networks that have little to no infrastructure. Lastly, in Section 6, we conclude with a summary of the survey paper.

2 Software-Defined Networking

Software-Defined Networking is a paradigm that decouples the control plane from the data plane. The data plane consists of the standard data packets that are forwarded by the network device. The logic to accomplish this forwarding is still at the hardware level. The control plane represents packets that manage the network devices. The control functionality has been lifted out of the hardware and is now defined in software, allowing network devices to be managed on-the-fly. This simple abstraction provides much needed flexibility to a variety of networking environments.

2.1 General Motivation

Before SDN, networking environments both in industry and academia were hindered by proprietary solutions with scant alternatives. Devices from one vendor typically could not interoperate with devices from another vendor. As a result, it was a painful for network operators to create useful networking features and tools. This situation locked many businesses into a particular vendor's hardware and tools. Alternatively, they were forced to accept the tools offered by the vendors. However, this sad state affected more than business networks. Research institutions could not find easy ways to perform realistic network testing [McKeown08]. When research institutions have difficulty progressing, the whole field lags. However, with SDN's separation of the data and control planes, networking companies can use hardware from a variety of vendors and still implement policies that persist across all of them.

2.2 OpenFlow

OpenFlow is the most notable instantiation of Software-Defined Networking principles, via the decoupling of the data and control planes. OpenFlow's architecture requires forwarding devices, also referred to as OpenFlow switches, and a network controller — this architecture is pictured in Figure 1. The controller propagates forwarding rules into each switch's flow table, a construct that contains information necessary to maintain the OpenFlow specification [McKeown08]. This allows the forwarding devices to make decisions based on flows of traffic, a level of control that was difficult to achieve before the days of OpenFlow. The OpenFlow protocol is the standardized protocol for the southbound interface, which is the communication between the network devices (OpenFlow switches) and the controller. However, the northbound interface, which is the communication between the controller and the network applications, has no standardized interface.

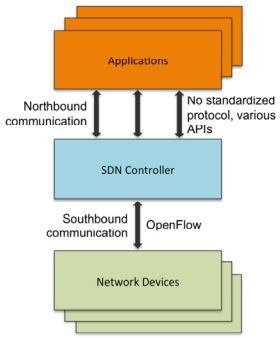

Figure 1: The OpenFlow Architecture

Each entry in the flow table consists of the following items: match fields, counters, and a set of actions. The fields available to be matched against are a 12-tuple of packet header fields [Pfaff09], which are listed in Table 1. This list has been updated to allow IPv6 addresses and protocol fields, but this 12-tuple highlights the general fields that OpenFlow inspects. Counters are used to accumulate statistics on the different flows passing through the forwarding device. Lastly, the set of actions are simply instructions on how to handle packet flows that matched the specific criteria in the matching fields [Nunes14].

In other words, the OpenFlow switch compares the particular fields of every packet against the match fields defined in the device's flow table. If the packet matches a particular flow entry based on the match fields, it is forwarded under the conditions of that rule and the appropriate counters are incremented. If no rule exists, OpenFlow follows the rule held in the table-miss flow entry, which defines an action for packets that do not match a particular flow entry [Nunes14]. Every flow table has a table-miss flow entry; the controller needs to know how to handle every packet. A common action for the table-miss flow entry is to send the first packet of the unknown flow to the controller. In response, the controller will create a new rule for the traffic flow with that specific 12-tuple, following the control logic that is programmed into it. The controller propagates this new flow entry to all OpenFlow switches so they can add the entry to their flow table. As a result, each switch will be able to handle a subsequent flow that matches those fields, without the need for further controller intervention.

Table 1: OpenFlow Match Fields

Ingress port into OpenFlow switch	Ethernet source address	Ethernet destination address	Ethernet segment type
VLAN identifier	VLAN priority	IP source address	IP destination address
IP protocol	IP type of service	TCP/UDP source port	TCP/UDP destination port

Though they technically are not a part of OpenFlow, NOX and FlowVisor are tools that are very commonly used with OpenFlow and are important to note. NOX is a network operating system that roughly acts as the network controller in the OpenFlow framework. A NOX controller has a global view of the network and allows management and control applications to be run with ease. FlowVisor is a special controller that allows for virtualization in networks. FlowVisor easily achieves traffic isolation and other nice properties that were quite difficult to achieve before SDN. These tools are repeatedly used throughout the research surveyed in this paper.

2.3 Common Misconceptions

There are a some notable misconceptions about OpenFlow outlined by Feamster, Rexford, and Zegura in [Feamster13] that are crucial for a nuanced understanding of OpenFlow. Particularly, there are two non-obvious points presented by the authors:

1. The first packet in every unmatched flow does not have to be sent to the controller. Some performance conscious OpenFlow networks do not send smaller flows to the controller, as they are not worth the controller's or the network's time. Instead, they are handled separately, i.e., dropped or sent to a different network device. In this scheme larger flows are still forwarded to the controller to prompt a new rule.
2. There can be more than one controller in an OpenFlow network. The logically centralized controller does not have to be physically centralized. Some systems use a set of distributed controllers operating as one logical controller. This is particularly important when the performance cost of forwarding to one centralized controller would be too high.

It is obvious that the first point is important for performance critical networks, which can be seen in both dense and light infrastructure networks. Overloading the controller should be avoided in networks with only one controller, and this is a way to reduce traffic destined for the controller. The second point is also seen in both dense and light infrastructure networks. Distributed controllers incur more overhead in wireless networks, but allow for more redundancy and reliability. Now that we have discussed the background material, we can dive into the focus of the paper: SDN in wireless networks.

3 Wireless SDN Techniques

Because OpenFlow was developed with wired networks in mind, it took a few years after its inception to apply SDN to wireless networks. However, over the past three years, a number of papers have laid the foundation of applying SDN techniques to wireless networking. Throughout these papers, there are a number of techniques that permeate the different applications of SDN to wireless networks. We discuss these techniques here in a more general sense, before visiting their specific applications in Section 4 and Section 5.

3.1 Slicing

Slicing is a technique that separates traffic flows into separate subspaces. These subspaces, or slices, are managed by a controller and share network resources [Feamster13]. Much like virtual memory for processes in an operating system, each slice has the illusion of its own distinct network resources. In actuality, many slices can be running on the same hardware. When combined with per-flow rules, this abstraction can isolate different slices of the network. This has remarkable implications towards network experimentation and network security, though the security issues are outside of the scope of this paper.

Chaudet et al. have a fantastic breakdown on the details of slicing. Specifically, they compare two strategies: allocating the same number of slices as there are channels available to the controller, and allocating more slices than there are available channels [Chaudet13]. The authors consider a controller that has the three independent 802.11 channels as a motivating example. The controller could decide to only allow for three slices, each of which is allocated an entire channel. This is by far the simpler

option. If the controller were to allocate more slices than there were available channels, things become more complicated. Now the traffic from slices sharing a channel must be multiplexed together, which involves guard intervals or fine-grained timing, depending on if time or frequency division multiplexing is used [Chaudet13]. If another multiplexing technique is employed, e.g., statistical multiplexing, there are even more complications.

3.2 Control Strategies

As mentioned in the previous section, OpenFlow can have one centralized controller or many distributed controllers. There are positives and negatives to each approach. With a centralized controller, global network state only needs to be maintained on one controller. As a result, no controller-to-controller communication strategy is necessary. In the OpenFlow architecture, there is no standardized communication protocol between controllers, so this is a reasonable option. Having one controller is also the cheapest option, which makes it a more feasible approach for network operators with less capital. However, if the controller fails, there is no other controller available; unmatched flows sent to the failed controller are essentially dropped. If this controller performs any path management for the network, the paths at the time of failure will remain until the controller comes back online. This is particularly important in the wireless setting, which has much more variability than the wired networks OpenFlow was designed for.

In general, it seems more intuitive to have distributed controllers throughout a wireless network. Because unmatched packets can be routed to the closest controller, it is a shorter trip for the packet to be forwarded to the controller. If there are a lot of unmatched packets, the traffic intensity can be portioned to each respective controller, ensuring that the SDN wireless network is processing packets as quickly as possible. If a controller fails, unmatched traffic can be sent a nearby controller instead. However, there are a few reasons this is not the most pervasive option. It is expensive to have multiple controllers on a network, which makes it less attainable for networks in emerging markets — unfortunately, these networks would benefit the most from distributed controllers. A comparison between the distributed and centralized approaches is summarized in Table 2. Additionally, there is no standardized protocol for communication between controllers. It is the network operator's duty to ensure these controllers can properly communicate. However, this introduces a further problem of whether the global state should be maintained through all controllers. If each controller needs the same global view of the network, the controllers must communicate any topology changes. On the other hand, if they are allowed to have varying views of the network, there will be less control traffic sent out, but some controllers could have a stale view of the network. As with most tough questions, the answer depends on the environment in which wireless SDN techniques are being applied.

Table 2: Comparison of Centralized and Distributed Controller Strategies.

	One Controller (Centralized)	**Many Controllers (Distributed)**
Cost	✓	
Complexity	✓	
Reliability		✓
Response Time		✓
Amount of Control Traffic	✓	

As a brief aside, different systems approach controller failure in different ways. Some systems use common routing protocols to carry data if the controller fails [Detti13]. The system still operates and

new routing decisions can be made, just not on a per-flow basis and without global knowledge of the network. Others simply use a backup controller when the other controller fails, and switch from the failed controller to the backup controller when a failure is detected [Nguyen12]. Others simply wait for the controller to restart, which is not an uncommon approach in networking [Suresh12a].

3.3 Traffic Engineering

With the ability to perform flow-based routing, many traffic engineering problems become easier to tackle. OpenFlow provides a level of generality that had not previously existed. As a result, it can be used to route flows based on any arbitrary values in the match field of the flow entries. The most common application of traffic engineering in SDN is load balancing. Gathering data from the counters in the forwarding devices, the controller can form statistics to gauge how heavily the device is utilized. If this meets a certain threshold, it is intuitive that the controller could propagate a new rule to relieve that link's load, distributing the excess load to other routes. An example of load balancing is pictured in Figure 2. The parameter rho represents the traffic intensity of the link. The topology is in steady state, i.e., the network load is fully balanced, when an influx of traffic arrives and is routed through node 3 to node 4. However, the 1-to-3 and 3-to-4 links are unfairly overloaded, as much of the traffic is being routed to node 4. There is no reason the traffic cannot go through the other side of the topology as well. As a result, the SDN controller load balances the wireless network, avoiding congestion. It is intuitive that this traffic engineering approach could extend to support energy-aware routing [Costanzo12]. By measuring the energy output rather than the device load, the controller could easily route around network devices that could be sleeping. Some wireless network operators employ energy aware routing when the network utilization is not very high, but when the utilization increases they switch to a load balancing strategy.

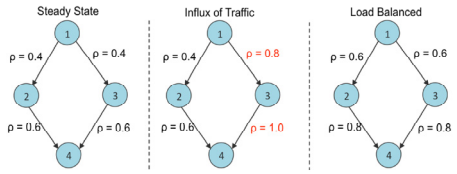

Figure 2: Load Balancing

The techniques highlighted in this section are used throughout the research surveyed in this paper. Each technique provides more flexibility, which is the hallmark of SDN. Wireless networks needed flexibility badly, so the application of SDN to them is intuitive. However, we can see that different wireless network environments call for different ways SDN can be applied. Broadly, there are two spheres that represent these differences: dense infrastructure and light infrastructure wireless networks. In the next two sections, we highlight the research that applies SDN to each of these network environments.

4 SDN in Infrastructure-Heavy Wireless Networks

OpenFlow was originally developed for use in campus networks. Over time, however, it has spread throughout wireless networking, touching many different sub-disciplines. In this section and the next, we highlight the current literature applying SDN to these fields. By the end of these sections, the techniques discussed in the last chapter should be quite familiar to the reader.

4.1 General Developments

Software-Defined Networking's most significant application to wireless networks is infrastructure-based wireless networks, e.g., wireless local area networks (WLANs) or cellular networks [Nunes12]. As with most networks, proprietary equipment and stale business models drastically hinder mobile and wireless networks [Yap10]. However, decoupling the control plane from the data plane could free more capacity and establish competition between different providers. Using OpenFlow, NOX, and FlowVisor, Yap et al. created the OpenFlow Wireless deployment, specifically testing the network with two handover algorithms between WiFi and WiMAX. Both mobility handovers performed significantly better than the hard handover. However, the notable aspect of the research was not the handover algorithms themselves. Rather, it is notable that for a class project, students designed, implemented, and tested these handover algorithms in less than a month. SDN has created a wireless environment that not only allows for slicing and better mobility handovers, but is so easy to use a non-expert can easily tweak the network in important ways. Also, the OpenFlow Wireless deployment allowed production networks to be sliced, so network testing could occur on the same equipment as the live network. This allows for much greater flexibility in testing, and is particularly helpful for research institutions.

The CROWD project, which applies SDN to "extremely dense wireless networks", provides potential solutions for both WLANs and cellular networks [Ali-Ahmad13]. The authors argue that incremental technology updates will not stand up to the increasing demands from telecommunications networks, and rather, we must revise the architecture of our networks as a whole. Dense wireless networks currently face high energy consumption, higher interference, and frequent handovers for mobile devices. As a result, the paper proposes an architecture that could reduce intra-system interference and the cost of mobility handovers. In order to reduce intra-system interference and increase capacity, Ali-Ahmad et al. propose tweaking certain 802.11 parameters — AIFS, CWmin, CWmax, and TXOPmax — for each base station. The SDN controllers would be able to dynamically set these parameters as the networks operated, allowing less interference to occur. In a sense the controllers smartly scale the transmission power levels to minimize interference. To further reduce intra-system interference, the authors suggest using the Almost Blank Subframe paradigm. This strategy selects and mutes subframes that induce too much interference [Pauli11]. For the mobility handoff, Ali-Ahmad et al. apply a simple tunneling approach to their newly proposed architecture. Their handoff algorithms were designed such that certain steps can be taken in parallel. It should be noted that applying this network architecture would require a number controllers in a different configuration than today's standards. As a result, it might only be useful to companies deploying a new wireless network with high AP density.

4.2 Wireless Local Area Networks

Only a small number of researchers have particularly looked to apply SDN to wireless local area networks (WLANs). The most prominent application to WLANs has been the SDN framework named Odin. Odin provides SDN-level programmability to enterprise WLANs, which typically support more flexibility and richer services than normal WLANs [Suresh12b]. As with other wireless networks, enterprise WLANs were stricken by proprietary solutions. However, this is exacerbated by the number of services an enterprise network needs to offer, including: authentication, authorization, accounting, policy management, dynamic channel reconfigurations, mobility management, interference management, and load balancing [Suresh12a]. One can imagine needing expensive, proprietary middleboxes for each service desired. Additionally, clients choose to associate with APs in WLANs, which makes it difficult to control access and provide isolation. To meet these goals, the authors created an abstraction called light virtual access points (LVAPs), which are essentially per-client access points. Suresh highlights that each client connected to an Odin network only sees its own

personal AP, no matter what physical APs are in the surrounding environment [Suresh12a]. LVAPs are an alternative to FlowVisor for SDN-based network virtualization. It is no surprise that the programmability and vendor-neutral properties of the network resulted from using SDN. However, the LVAPs provide a powerful addition: they allow clients to undergo quick and cheap re-associations. These are so effective that using LVAP re-association for a mobility handover yields no throughput degradation.

4.3 Cellular Networks

The application of SDN to mobile networks has unearthed scalability issues that were not encountered before. Particularly, mobile networks provide for a large number of subscribers who are frequently mobile, and the operators need to measure and control their traffic to provide services consistent with their business goals. At the forefront of the research, Li et al. propose the application of SDN to LTE, which provides not only the commonly known flexibility to their network, but also provide the fine-grained policy control necessary for mobile network operators [Li12]. The most fundamental advantage of applying SDN to mobile networks is the ability to distribute the data plane over multiple, cheaper network switches, as opposed to the packet gateways (P-GW) in the network. Previously, all traffic was required to go through P-GW. These P-GWs are extremely expensive (on the order of millions of dollars), so network operators frequently overloaded them. In other words, the operators would not buy as many P-GWs as they needed to save money, but this degraded the service quality of the network. However, with SDN operators can now buy a large number of cheaper devices from different vendors and distribute them, allowing for better traffic engineering. The authors also argue this approach allows for mobility between different cellular providers, when the provider of the customer's choice does not have service in an area. With an SDN architecture, it would not be difficult to perform the measurements needed to properly bill the guest's mobile network operators. Lastly, because the controllers would have a global view of their base stations, they could be used for fine-grained subcarrier coordination to reduce inter-cell interference [Li12].

We see that SDN provides flexibility in a variety of infrastructure-heavy wireless network settings. In the infrastructure-heavy setting, there is much more of an industry bend. As a result, SDN's flexibility allows these dense wireless network operators to have a variety of vendor equipment, improve the network's latency, and perform cheap handovers between different wireless networking technologies.

5 SDN in Light or Zero Infrastructure Wireless Networks

As mentioned in the previous section, SDN has been applied to many sub-disciplines. Some of these applications are less obvious than the previous section's, e.g., rural wireless networks. In this section we present the application of SDN in wireless networks with little-to-no infrastructure.

5.1 Wireless Mesh Networks

Wireless mesh networks (WMNs) are a cost-effective way to expand the coverage area of a WLAN [Dely13]. A typical wireless mesh network will have a small number of nodes connected to the Internet; the rest of the nodes are connected to one of these nodes through a multi-hop path in the mesh topology. The topology allows for reduced outside connectivity, which is how WMNs save money. SDN is particularly applicable to WMNs because of the routing decisions necessary for inner nodes to communicate with the Internet. Broadly, SDN provides a global view at the controller, and

per-flow routing capabilities to allow for traffic engineering and reliability. It is intuitive that these traits are important in a WMN.

Dely et al. applied SDN to WMNs by using OpenFlow and NOX controllers. To achieve virtualization, they used different SSIDs [Dely11]. Their routing strategy fell out of the decoupling of the data and control planes. By separating the data and control planes, the authors could use different routing strategies for the traffic on each plane. Particularly, the VLAN carrying the data traffic was routed on a per-flow basis by OpenFlow Mesh Nodes. The VLAN carrying the control traffic used OLSR, a standard multi-hop routing protocol. However, this paper received mixed results in performance tests. It appears the amount of control traffic for OpenFlow scaled linearly with the number of rules. This implies there is a middle ground between the dynamicity of the network and the overhead incurred by control traffic. The more dynamic the mesh network is, the more control traffic it must send out, and the more overhead introduced into the system. Additionally, Dely argues that association is not as straightforward in the case of a WMN [Dely13]. Typically, users connect to the AP with the strongest signal. However, in a WMN, load balancing can be crucial to avoid congestion, but the previous strategy will overload stronger APs. As a result, he proposes allowing the controller to device which AP a client should associate with, in hopes that it will be a more informed decision.

Detti et al. take a more balanced approach to using SDN strategies in WMNs. They acknowledge that the OpenFlow capabilities are inherently valuable, but they should be leveraged in a more careful nature. The previous research did not address what to do when the OpenFlow controller fails. In this project, they maintain the single controller strategy — Openflow still provides the traffic engineering capabilities that are necessary in a WMN. Optimized Link State Routing (OLSR) always routes the control traffic between the OpenFlow switches and controller; similarly, OpenFlow routes the data traffic. However, OLSR takes on a bigger role. In the event that the OpenFlow controller is unreachable, OLSR routes the data traffic until it notices that the OpenFlow controller is back up [Detti13]. As a result, the network can still adapt even if the controller has failed. This is important, because situations where the controller is unreachable are not necessarily rare. The authors did not use VLANs like the last study, as they chose the control traffic to be carried out-of-band, which VLANs are not capable of doing. This software-defined WMN performed well in the evaluation phase, and did not have too much control traffic. As a result, it successfully addressed the reliability concerns that a WMN has, and took advantage of the flexibility that SDN offers.

5.2 Home Networks

Applying SDN to home wireless networks is a relatively new pursuit. There is a fundamental difference between the tools of the Internet, which were largely designed with enterprises in mind, and the tools necessary for non-experts to properly manage their home networks [Mortier12]. Home networks are becoming increasingly heterogeneous, as home users typically connect many different devices to one wireless router. This is particularly because devices have very different use cases, such as entertainment, work, and communication [Yiakoumis11].

There are two ways SDN is applied to home networks [Mortier12, Yiakoumis11], and they both take a similar approach, seeking to provide simpler management, per-flow control, traffic isolation, and a more intuitive management interface. Both groups use NOX controllers and OpenFlow as the southbound protocol. However, there are differences, particularly in how they achieve traffic isolation and the provided management schemes. Mortier et al. use custom DHCP and DNS implementations to provide fine-grained control of the home network. Specifically, they used DHCP to control the associations to their network, which naturally extended itself to enforce temporary leases and guest

addresses. The authors modified DNS to manage connectivity for each device separately, enforcing domain restrictions set by the home network operator [Mortier12]. OpenFlow and NOX provided the flow-based management as desired. However, what makes this paper unique is its distinct focus on the user. They created a network control touchscreen, similar to a thermostat, in order to allow the home user to have intuitive control of their network. If they want to allow a device to associate with the network, they drag the device's tile to right of the screen. If network operators want to reject the device, they drag it to the left of the screen. These gestures are natural for most people, particularly given the ubiquity of smart phones and tablets.

While the other approach was user focused, Yiakoumis et al. pursue a direction that is more focused on the business implications of applying SDN. Separate slices of the network can be provided to the different stakeholders. For example, if a mobile network operator does not provide enough signal to an area, it would be quite convenient to use a slice of the home wireless network, provided a previous agreement was established. Similarly, bandwidth intensive Internet services could greatly benefit from their own slice, allowing them to optimize the settings for their use case, e.g., Netflix optimizing their slice of a home network for video transmission [Yiakoumis11]. Slicing could allow the non-expert home network operator to be removed from the maintenance role, as the companies can manage their own pre-defined slices. Because SDN techniques are applied to the network, this notion can be furthered: companies could remotely control a large group of consumers' networks. It is easy to see how this strategy is appealing for everyone. Companies have more fine-grained control over their slices to maximize the performance. A new business model would allow network administration companies to manage many home networks remotely.

Software-Defined Networking has also been applied to wireless personal area networks (WPANs). Specifically, Costanzo et al. apply them to low-rate WPANs (LR-WPANs). The simplifications and management features that SDN provides are attractive to any type of networking environment. The authors pursued SDN for the simplifications, flexibility, and management facilities it is touted to provide. Specifically, they look to reduce the energy consumption in their software-defined LR-WPANs, under the assumption that periodically turning off the interface of generic devices would save energy [Costanzo12]. The authors also argue that matching must be more flexible for LR-WPANs than it is required for other networks. Specifically, they argue for matching against a threshold, as mentioned in the OpenFlow section of the paper, to provide that extra flexibility.

5.3 Rural Wireless Network Operators

Perhaps the most challenging application of Software-Defined Networking is wireless Internet service providers (ISPs) in emerging markets, as they are markedly resource constrained. In [Hasan12], the authors label these constrained wireless ISPs with the term rural wireless network operators (RWNOs). RWNOs must scrape together a working wireless network from very different devices, with the hope that the network can withstand the harsh environment around it, while still delivering acceptable service. The paper highlights problems of a real RWNO from the Indian Himalayas, called AirJaldi. Some major issues that RWNOs face include:

1. People would accept network administrator positions with AirJaldi, stay through training, and then — once they felt sufficiently skilled — would leave for a higher paying job at another company.
2. The network is easily congested. If people are using an absurd amount of bandwidth, they need to be limited. Similarly, if storms or other adverse conditions knock out one of the RWNO's upstream providers, the RWNO must quickly reroute the traffic through its other upstream

providers. It is important to note RWNOs do not have many upstream providers, e.g., AirJaldi has two. This substantial influx of traffic must be dealt with. As a result, RWNOs are heavily reliant on traffic engineering, without which they would be crippled by congestion.

3. Because the networks relied on such a variety of vendor hardware, many of the tools supplied by vendors did not interoperate with each other. This created a chaotic set of tools for each RWNO, resulting in solutions that were often heuristic and could not be easily shared with other RWNOs. Essentially, the only option for RWNOs was to learn the hard way.

However, the authors argue that applying SDN principles to RWNO's networks would alleviate each of these concerns. To counter the first issue, they argue that SDN would actually allow RWNOs to alter their business model. By decoupling the control and data plane, RWNOs can decouple the physical network construction from the network administration [Hasan12]. The particular benefit of this decoupling is that the network administration can be done remotely. RWNOs could then focus on training employees to deploy their complex physical infrastructure, which requires substantial domain specific knowledge. The network administration could come from a larger company or ISP, who have substantial experience in the traffic engineering. This remote network administration strategy mitigates the issue of network administrators leaving the RWNO after they have been trained — now the RWNO's network administrators work for the companies they would have left the RWNO for in the first place!

Other SDN techniques for wireless networks could further help these constrained RWNOs. At times when congestion is not an issue, the controller could route flows based on minimizing power consumption. Slimming down the operating expenses even by a slim margin would provide valuable savings for a RWNO, allowing them to reinvest the money in their infrastructure or company. Substitution networks [Venmani12], which are temporary networks used to relieve overloaded base network resources, could be a valuable investment for some RWNOs. If there are patterns inherent to the congestion, a substitution network could provide more versatility than upgrading capacity in one spot.

As promised, wireless networks with little-to-no infrastructure are vastly different than ones with dense infrastructure. However, many of the valuable features SDN brought were the same for both networks. However, there are some stark differences. Allowing different vendor equipment is a huge benefit for wireless networks in emerging markets; they can interoperate whatever equipment they come across, instead of needing to save up for and purchase new equipment. Similarly, applying SDN principles to home networks and Rural Wireless Network Operators have the potential to fundamentally change their business models. Lastly, the traffic engineering is inherently valuable in all of the cases in this section.

6 Summary

Software-Defined Networking is a valuable abstraction, and it helps solve many woes when applied to wireless networks. The general benefits of SDN make a strong argument: it can run on nearly any vendor equipment, can provide fine-grained control of the network, and has unparalleled ease-of-use. In dense wireless networks, the vendor equipment improvement is inherently valuable, because these pieces of equipment are deployed on such a large scale. To be free from vendor lock-in relieves a tremendous burden for a business. Similarly, SDN can minimize many of the sources of interference and other network inefficiencies, by allowing APs to coordinate with each other. They can agree on acceptable power rates. They can load balance and distribute network associations so one network is not overloaded. Similarly, these benefits are very valuable to networks without much infrastructure.

SDN can provide wireless mesh networks with valuable load balancing techniques, controller redundancy, and cheap equipment. The ease-of-use it can provide infrastructure-less networks is very important, as many times these network operators are resource constrained and do not have a large number of employees. It is true that different wireless networks have different priorities. However, SDN is flexible enough that it can provide valuable features while maintaining those priorities. It truly is a groundbreaking technique, and will continue to make waves in the world of networking.

References

1. [Nunes14] Bruno Astuto A. Nunes et al., "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," IEEE Journal of Communications Surveys & Tutorials, Vol. PP, Issue 99, pp. 1-18, 2014. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6739370
2. [Yap10] Kok-Kiong Yap et al., "Blueprint for Introducing Innovation into Wireless Mobile Networks," Proceedings of the 2nd ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures, pp. 25-32, 2010. http://dl.acm.org/citation.cfm?id=1851404
3. [Kolias13] Christos Kolias et al., "OpenFlow-Enabled Mobile and Wireless Networks," Open Networking Foundation Solution Brief, pp. 1-13, 2013. https://www.opennetworking.org/solution-brief-openflow-enabled-mobile-and-wireless-networks
4. [Dely13] Peter Dely, "Architectures and Algorithms for Future Wireless Local Area Networks," Doctoral Thesis, Karlstad University, pp. 1-256, 2013. http://www.diva-portal.org/smash/get/diva2:563553/FULLTEXT01.pdf
5. [Suresh12a] Lalith Suresh Puthalath, "Programming the Enterprise WLAN: An SDN Approach," Master's Thesis, Technical University of Lisboa, pp. 1-86, 2012. http://lalithsuresh.files.wordpress.com/2011/04/lalith-thesis.pdf
6. [Detti13] Andrea Detti et al., "Wireless Mesh Software Defined Networks," IEEE Conference on Wireless and Mobile Computing, pp. 89-95, 2013. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6673345
7. [Dely11] Peter Dely, Andreas Kassler, and Nico Bayer, "OpenFlow for Wireless Mesh Networks," IEEE Conference on Computer Communications and Networks, pp. 1-6, 2011. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6006100
8. [Suresh12b] Lalith Suresh et al., "Towards Programmable Enterprise WLANs with Odin," Proceedings of the 1st Workhsop on Hot Topics in Software Defined Networks, pp. 115-120, 2012. http://dl.acm.org/citation.cfm?id=2342441
9. [Li12] Li Erran Li, Z. Morley Mao, and Jennifer Rexford, "Toward Software-Defined Cellular Networks," European Workshop on Software Defined Networking, pp. 7-12, 2012. https://www.cs.princeton.edu/~jrex/papers/ewsdn12.pdf
10. [Feamster13] Nick Feamster, Jennifer Rexford, and Ellen Zegura, "The Road to SDN: An Intellectual History of Programmable Networks," pp. 1-13, 2013. http://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf
11. [Yiakoumis11] Yiannis Yiakoumis et al., "Slicing Home Networks," Proceedings of the 2nd ACM SIGCOMM Workshop on Home Networks, pp. 1-6, 2011. http://yuba.stanford.edu/~nickm/papers/Homenets2011.pdf
12. [Venmani12] Daniel Philip Venmani et al., "Substitution Networks Based on Software Defined Networking," Fourth International ICST Conference on Ad Hoc Networks, pp. 242-259, 2012. http://link.springer.com/chapter/10.1007%2F978-3-642-36958-2_17

13. [Chaudet13] Claude Chaudet and Yoram Haddad, "Wireless Software Defined Networks: Challenges and Opportunities," IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems, pp. 1-5, 2013. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6685237
14. [Ali-Ahmad13] Hassan Ali-Ahmad et al., "An SDN-based Network Architecture for Extremely Dense Wireless Networks," IEEE Conference on SDN for Future Networks and Services, pp. 1-7, 2013. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6702534
15. [Hasan12] Shaddi Hasan et al., "Enabling Rural Connectivity with SDN," University of California, Berkeley Technical Report, 2012. http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-201.html
16. [Nguyen12] Xuan Nam Nguyen, "Software Defined Networking in Wireless Mesh Networks," Master's Thesis, University of Nice Sophia Antipolis, pp. 1-37, 2012. http://www-sop.inria.fr/members/Xuan-Nam.Nguyen/Publications/Internship%20Report_NGUYEN%20Xuan%20Nam.pdf
17. [Mortier12] R. Mortier et al., "Control and Understanding: Owning Your Home Network," Fourth International Conference on Communication Systems and Networks, pp. 1-10, 2012. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6151322
18. [Costanzo12] Salvatore Costanzo et al., "Software Defined Wireless Networks: Unbridling SDNs," European Workshop on Software Defined Networking, pp. 1-6, 2012. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6385039
19. [Pauli11] Volker Pauli and Eiko Seidel, "Inter-Cell Interference Coordination for LTE-A," Novel Mobile Radio Research, pp. 1-8, 2011. http://www.nomor.de/uploads/1d/19/1d196a493af5511cc92466089924cc5c/2011-09-WhitePaper-LTE-A-HetNet-ICIC.pdf
20. [McKeown08] Nick McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks," ACM SIGCOMM Computer Communication Review, Vol. 38, Issue 2, pp. 69-74, 2008. http://archive.openflow.org/documents/openflow-wp-latest.pdf
21. [Pfaff09] Ben Pfaff et al., "OpenFlow Switch Specification," OpenFlow Consortium, pp. 1-42, 2009. http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf

List of Acronyms

AP	Access Point
DHCP	Dynamic Host Control Protocol
DNS	Domain Name System
IP	Internet Protocol
LTE	Long Term Evolution
LR-WPAN	Low-Rate WPAN
LVAP	Light Virtual Access Point
OLSR	Optimized Link State Routing
P-GW	Packet Gateway
RWNO	Rural Wireless Network Operator
SDN	Software-Defined Networking
TCP	Transmission Control Protocol

UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WPAN	Wireless Personal Area Network

Last Modified: April 30, 2014
This and other papers on current issues in Wireless and Mobile Networking are available online at
http://www.cse.wustl.edu/~jain/cse574-14/index.html
Back to Raj Jain's Home Page