Software-Defined Networks and Their Uses

Olivia Lazarchick  
Department of Computer Engineering  
University of Pittsburgh  
Pittsburgh, PA  
[oll12@pitt.edu](mailto:oll12@pitt.edu)

Avery Peiffer Department of Computer Engineering University of Pittsburgh Pittsburgh, PA [aep65@pitt.edu](mailto:aep65@pitt.edu)

Elissa Wilton  
Department of Computer Engineering   
University of Pittsburgh  
Pittsburgh, PA  
[eaw98@pitt.edu](mailto:eaw98@pitt.edu)

*Abstract*—*Software-defined networks (SDNs) are an increasingly popular approach to network management that use dynamic network configurations to increase efficiency. SDNs draw on advances in cloud computing to create centralized networks that are more flexible than traditional network architectures. This report examines the motivation behind the rise of SDNs, including their usefulness for supporting Internet of Things (IoT) devices. The report will outline the technology of SDNs and compare their performance to traditional network architectures across various applications. Finally, the report will discuss the limitations of SDNs and the technology’s future.*

Keywords—software-defined networking, control plane, data plane, management plane, application programming interface, Internet of Things

# Why Software-defined Networks are Needed

*A. Traditional Networking and its Shortcomings*

The first large scale, packet-switched computer network was the ARPANET, designed in 1969 by the U.S. Department of Defense's Advanced Research Projects Agency. ARPANET was an incredible technological achievement for its time, using a technology known as the Interface Message Processor (IMP) to connect host computers to the telephone circuit used by the network [1]. These IMPs were incredibly simple compared to the capabilities of routers today; their only functions were to store and forward [2]. While it would eventually be outclassed by more modern networking technologies, ARPANET represented a considerable leap in the realm of packet switched networking.

Despite the considerable amount of networking innovation that has occurred in the decades following ARPANET, conventional networks in the present share one key similarity with ARPANET: they are predominately hardware based [3]. Traditional networking is implemented as a series of devices with very specific functions. Each device must perform its own function well to support the network as a whole. As a result, network architects have been heavily incentivized to implement these devices as fixed hardware constructs to increase the network's speed. In this way, traditional networks implement both a control plane and a data plane in hardware; the data plane forwards packets through the network, while the control plane uses routing protocols to find pathways to forward the packets [4].

The downside of traditional networks is their rigidity; since both the control plane and data plane must be implemented in hardware, traditional networks do not offer architects the opportunity to install new or experimental protocols to improve the management of resources in the network [4]. This inflexibility came with few consequences until recently, as customers had very straightforward networking needs. However, the proliferation of cloud services in recent years requires additional storage and bandwidth, incentivizing businesses to more directly manage how their computing resources are allocated [3].

*B. Cloud Computing and the Internet of Things*

Cloud computing has emerged as a powerful means by which users can directly share resources. The cloud refers to servers that are accessible remotely, over the internet, instead of on local machines. The cloud also includes the software and databases that run on said servers. By utilizing cloud computing, users can store and access data from a remote server, without the responsibility of managing the physical servers [5]. This results in more robust and efficient applications, as there is virtually no limit to an application’s data storage.

One specific application of cloud computing is in devices connected to the Internet of Things (IoT). An IoT device is an embedded device that has internet connectivity, as opposed to being an entirely isolated unit [6]. IoT devices can be any external device that is connected to the internet; common examples include kitchen appliances, home security systems, and health monitors that are worn [6]. As might be expected, IoT devices generate massive amounts of data, putting “immense pressure on the internet infrastructure” [7]. However, in what can be described as a “perfect union”, cloud computing provides convenient processing capabilities for the data collected by IoT devices [7]. Instead of having to perform both data collection and data analysis, IoT devices can be built with data collection as their sole focus; the data is then sent to remote cloud servers, where it is actually processed.

The demand for IoT devices continues to grow. In 2019, 7.6 billion IoT devices were connected to the cloud. Reasonable forecasts estimate that by 2030, the number of connected devices will increase more than threefold to 24.1 billion [8]. As these cloud environments continue to gain usage, either with traditional devices or with IoT devices, it is critical that they are flexible and adaptable to the needs of many different users, especially now that cloud computing has become a fixture of business. Due to their inflexibility, traditional networking does not meet the needs of many businesses aiming to have direct control of their network infrastructure. Software-defined networks, on the other hand, lend themselves perfectly to this issue.

*C. Software-Defined Networking*

Software-defined networking (SDN) technology is better suited to handle the growing needs of modern networks, as compared to its hardware-based counterparts. Software-defined networks have an infrastructure controlled by software, allowing for higher flexibility and interoperability with lower operating costs. By decoupling the control and data plane, the network can connect directly to applications through application programming interfaces (APIs) [3]. The direct API access allows network programmers the freedom to interface and reconfigure the network and its components, creating a flexible network that can adapt to the unique requirements of each individual user.

The key benefits of software-defined networks arise from the use of network virtualization. This is the process of creating a virtual network by abstracting network resources that have traditionally been hardware-based to software-based. By implementing software-defined networks, businesses can support a more efficient use of their network resources to support virtual machines, achieved through OpenFlow virtual switches. This process allows for greater flexibility within resource allocation because virtual machines can be dynamically shared across virtual switches, whereas before they were locked to a static IP address [13]. Aside from creating adaptable networks, this technology also lowers operating costs and improves data efficiency.

Another notable advantage of SDN is centralized traffic management. A software-defined network has a central control unit that stores all of the network information and application requirements. This control unit can collect real-time network status information and direct traffic in accordance with that knowledge. Whereas traditional networks are fixed in resource allocation, the SDN controller can adjust traffic flow to best meet the requirements of the network quality of service (QoS) [10]. This makes for a flexible network that can meet increasing traffic demands.

SDNs give administrators more control within their network and encourages innovation. However, the technology is still young, and does not come without challenges. One of the greatest challenges is standardization. OpenFlow is popular, but it is not the only SDN standard [14]. Security is also of concern. The central control unit that software-defined networks implement is a defined point of weakness within the network. For software-defined networking to have a successful integration, there must be interoperability with legacy network devices, security protocols in place, and experts to give technical help [14].

# Main Objective

## Architecture

Software defined networks use the common computing best practice of separation of concerns to increase flexibility and interoperability. This is done through separation into three layers: the data plane, the control plane, and the management plane. The data plane consists of all the hosts and devices connected to the network. One of the goals of SDNs is to simplify these components into basic forwarding engines. The interactions between those engines are then defined by the control plane [11].

The control plane, or controller, is where the switching and routing functionalities lie in a software defined network. The control plane has a globalized view of the complete network, which makes it flexible. In contrast to traditional networks, SDNs can implement policies and protocols for the network as a whole or on an individual device through the external controller instead of being forced to treat each network device independently. This globalized view of the network allows for greater efficiency because with knowledge of the entire network, the control plane can choose a truly optimal network path as opposed to a locally optimal path, as is the case with traditional networks [11].

The management plane that exists in hardware-based networks is present in software-defined networks, as well. The management plane is a typically centralized collection of functions that ensures the network is running optimally by communicating with the network’s operational plane. The management plane is responsible for monitoring, configuring, and maintaining network devices. Within software-defined networks, the distinction between the management and control planes is not so clear-cut. But there are important distinctions. One of the biggest differences is how the planes interact with the networking device. While the control plane focuses mainly on the forwarding plane, the management plane has its focus set on the operational plane of the device. Another distinction is the timescale. The control plane needs to react quickly to changes in the network, requiring high-bandwidth and low-latency links. The management plane takes more time in its decisions, so the speed of data transmission is less of a concern [16].

An open-source interface called OpenFlow is the most widely used protocol for communication between the controller and the network devices. The OpenFlow architecture consists of the OpenFlow channel, connecting OpenFlow-enabled switches with one or more OpenFlow controllers [15]. One of the main features of OpenFlow, and part of what differentiates OpenFlow from traditional networking, is the use of flow and group tables to specify packet look-up and forwarding at the switches. The flow tables are a list of flows, which are essentially sets of packets matching certain specifications, along with instructions to handle each flow and statistics regarding the use of that flow table entry [15]. Possible instructions for a packet based on the flow it matches include packet forwarding, packet modification, pipeline processing, and group table processing [15]. Pipeline processing is when the instructions for a flow are to forward the packets to another table for further instructions and to allow information to be collected and shared between tables. Group tables also offer additional flow instructions and forwarding options such as broadcast or multicast. The OpenFlow controller manages the flow tables of the switches and makes decisions regarding the handling of packets that don’t match flow or group table entries [15].

## Network Flow Resolution

In addition to increasing the ability to optimize flow across a network by choosing global best paths instead of local best next steps, the global view possessed by the control layer in SDNs offers a promising structure to optimize multi-application quality of service; this controller takes application types and path availability into account across the entire network [17]. One idea of how this can be achieved is through two primary functions: the quality-of-service matching and optimization function (QMOF) and the path assignment function (PAF) [17]. QMOF uses a series of parameters about the user and application to calculate the optimal flow path for that service. It also calculates a hierarchy of next-best path options which allow controlled degradation of the service based on resource availability, known as the Media Degradation path (MDP) [17]. The MDP is sent to the path assignment function, which uses the OpenFlow structure to

maintain a database of all the flows in the network and knowledge on which flows are active. The PAF can then use this data to effectively integrate and adapt flows, aiming first for each flow’s optimal path and then, if that is not possible, attempts to implement subsequent options offered by the MDP. Once an optimal solution has been found for the multitude of services and application using a network, the PAF can use the OpenFlow Interface to that solution by configuring the flow tables on each of the network devices [17].

## Comparison to TCP/IP Network Structure

Layering in the TCP/IP framework allows for some flexibility in that each layer can be modified or improved without affecting the other layers. For example, upgrading an application doesn’t necessitate a change in the network’s IP address. However, the horizontal layout of traditional networks fails to implement this same philosophy. Instead, because routing decisions are distributed, integrating new network devices necessitates reconfiguration of other nodes [11]. This becomes increasingly laborious as networks change more often to meet user needs. Software-defined networks expand on this idea of layering by implementing centralized network management to allow flexibility in modifying the devices connected to a network. The isolation between the management, control, and data planes effectively adds additional layering that allows SDN enabled routers to act exclusively as forwarding devices instead of including the functionality of all three of those planes on each router, as is seen in traditional networks [11]. As explained in the previous section, this centralized network management also allows for more effective flow optimization compared to traditional networks. This is because decisions about routing are made from the control plane which has knowledge of the entire network instead of making routing decisions locally at each router.

# Evaluation

Our simulation highlights the property of SDN control planes to send packets according to the globally optimal path in a network. By contrast, since traditional networks are implemented entirely in hardware, the individual devices can only calculate the locally optimal paths for forwarding data. To highlight this, we created a simplified partial mesh network structure implemented in Java. The actual network, of routers and links, is populated by passing a text file into the program on the command line. Figure 1 below shows a visualization of one of the sample networks used to test the network. The command line interface also allows the user to run the shortest path calculation either as an SDN or a traditional network, after which the results can be compared.

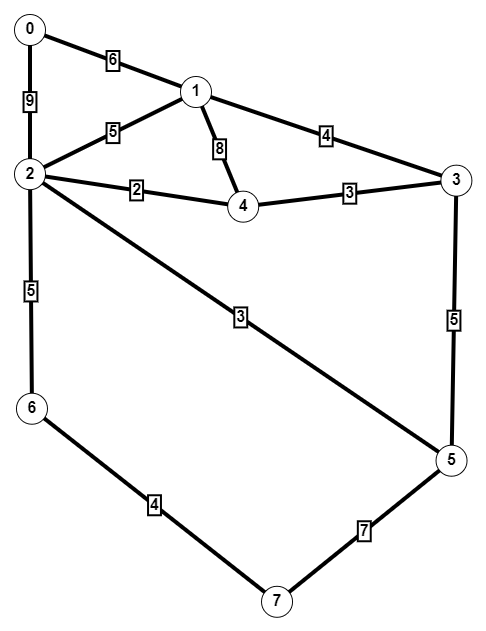


Figure 1: A visual representation of one of the small networks used for our simulation, created using Graph Online [12].

Put simply, the control plane of each network can be thought of as an algorithm trying to find the shortest path from the source node to the destination node. In our program, the hardware network finds the locally optimal path by finding the shortest path from the current node and advancing. For example, using the network structure in Figure 1, the shortest path from Node 0 to Node 7 would be calculated on a node-by-node basis from the perspective of the hardware controller. The SDN algorithm will instead use Dijkstra's shortest path algorithm to compute distances between nodes from a global perspective before resolving any paths. The SDN controller will conclude that, if the goal is to transmit data from Node 0 to Node 7, initially moving the data from Node 0 to Node 2 results in a shorter path than moving from Node 0 to Node 1. By contrast, the hardware network computes these distances from a more limited perspective; in this example, it will send data from Node 0 to Node 1 even though that eventually results in a longer path. The limited perspective of the hardware network also results in more dead ends, as it is not computing shortest paths with a global view of the network and thus may append to a path some nodes that are not useful in traveling to the destination. Though the units of length for the links are arbitrary, Table 1 shows the difference between path lengths computed by the hardware and software controllers for representative pairs of nodes in the network.

|  |  |  |  |
| --- | --- | --- | --- |
| Source Node | Destination Node | Length – hardware | Length – software |
| 0 | 7 | 25 | 18 |
| 4 | 5 | 5 | 5 |
| 5 | 1 | 12 | 8 |
| 0 | 3 | 10 | 10 |
| 6 | 4 | 16 | 7 |

Table 1: The lengths calculated using the separate algorithms implemented by the hardware and software controllers.

As Table 1 shows, the software network is not always better than the hardware network; for nodes that are closer together, the hardware network is more likely to calculate the true optimal path between source and destination. However, as the nodes become farther apart, the hardware controller is more likely to calculate an incorrect path between nodes. This specific simulation is obviously an exaggeration of the path-calculating issue, as hardware-defined networks do not consider only one node when finding the locally optimal path.

Perhaps a more accurate representation of the hardware network would be to implement Dijkstra's shortest path algorithm on a small number of nodes that do not cover the entire path from source to destination. However, the overarching point is still salient as it shows that SDNs will calculate more optimal paths for forwarding the data since the decoupled control plane can view the network in its entirety. The caveat with this entire experiment is that, by virtue of being implemented entirely in hardware, the hardware-based network will generally be faster than the SDN. In the cases where the hardware network is able to compute the true optimal path between source and destination, it will therefore yield higher throughput than the SDN controller. However, the important takeaway from this experiment is that the hardware controller has a very limited perspective on the network structure, which can cause it to make suboptimal decisions in path calculation.

# Conclusion

In conclusion, the flexibility of SDNs show great promise for the interconnected world in which we now live, especially with a drastic rise in IoT devices. Brad Casemore, research vice president for the International Data Corporation (IDC), states that “agility is the key attribute of digital transformation, and enterprises will adopt architectures, infrastructures, and technologies that provide for agile deployment, provisioning, and ongoing operational management. In a datacenter networking context, the imperative of digital transformation drives adoption of extensive network automation, including SDN” [9]. This quote shows the importance of SDNs in the present and future digital landscape.

However, SDNs are not the endpoint of networking innovation. While their flexibility offers a host of benefits to businesses aiming to control their network infrastructure, SDNs must sacrifice a nontrivial amount of speed to achieve this. It is entirely possible that SDNs will fall out of favor due to an increased emphasis on data rates; for example, as IoT devices become more sophisticated and require more data to function, hardware-implemented networks may become the more effective choice again. SDNs also suffer from a lack of mainstream focus; as stated in previous sections, the youth of SDNs means they suffer from a lack of standardization and underdeveloped security. Overall, SDNs simply present another option in customizing a network infrastructure based on an individual or organization’s specific needs.

##### References

1. H. Nelson, J. Mathis, and J. Lieb, “The ARPANET Interface Message Processor (IMP) Port Expander (PE).” [Online]. Available: <http://www.sonic.net/~warthman/images/SRI%20ARPANET-IMP.book.pdf>.
2. K. Hafner and M. Lyon, *Where Wizards Stay Up Late: The Origins of The Internet*. New York etc.: Simon & Schuster paperbacks, 2006.
3. “SDN Versus Traditional Networking Explained,” *IBM*. [Online]. Available: <https://www.ibm.com/services/network/sdn-versus-traditional-networking>. [Accessed: 13-Mar-2021].
4. N. Foster et. al, “Languages for Software-Defined Networks,” IEEE Communications Magazine, tech., 2013.
5. <https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=1283&context=jitim>
6. S. C. Mukhopadhyay, *Internet of things: challenges and opportunities*. Cham: Springer, 2014.
7. “The potential of IoT and Cloud Computing,” *IoT Tech Expo*, 31-May-2019. [Online]. Available: <https://www.iottechexpo.com/2019/05/iot/potential-of-iot-and-cloud-computing/>. [Accessed: 24-Mar-2021].
8. Jimmorrish, “Where does the growth in connected devices come from?,” *Transforma Insights*. [Online]. Available: <https://transformainsights.com/blog/where-does-growth-in-connected-devices-come-from#:~:text=Overall%2C%20IoT%20connected%20devices%20per,2019%20to%2013.24%20in%202030.&text=It%20is%20clear%20that%20certain,the%20total%20growth%20than%20others>. [Accessed: 24-Mar-2021].
9. M. Cooney, “What is SDN and where software-defined networking is going,” *Network World*, 16-Apr-2019. [Online]. Available: <https://www.networkworld.com/article/3209131/what-sdn-is-and-where-its-going.html>. [Accessed: 04-Apr-2021].s
10. *IEEE Xplore Full-Text PDF:* [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7496952>. [Accessed: 04-Apr-2021].
11. N. Zilberman, et. al, “Reconfigurable Network Systems and Software-Defined Networking,” *Proceedings of the IEEE, Volume 103, Issue 7I*, July 2015. [Online]. Available: <https://ieeexplore.ieee.org/document/7122247?denied=>. [Accessed 25-Mar-2021].
12. “Find shortest path,” *Русский*. [Online]. Available: <https://graphonline.ru/en/>. [Accessed: 04-Apr-2021].
13. Keith Kirkpatrick 4K Research & Consulting, “Software-defined networking,” *Communications of the ACM*, 01-Sep-2013. [Online]. Available: <https://dl.acm.org/doi/fullHtml/10.1145/2500468.2500473?casa_token=liLOSKIdgoEAAAAA%3AtvZFkSoAYmZ-0N0ZK3TkikKDquQ8cY6XmfrXIJl6YeIaNIdyi5Pi0rEZl-yFvaTKFMzqXU29r7Sf>. [Accessed: 04-Apr-2021].
14. “A Survey on Software-Defined Networking,” *IEEE Xplore*. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/6834762>. [Accessed: 04-Apr-2021].
15. M. Jammal, et al. “Software Defined Networking: State of the Art and Research Challenges,” *Computer Networks, Volume 72I,* 29-Oct-2014. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128614002588>. [Accessed 25-Mar-2021].
16. “doc: RFC 7426: Software-Defined Networking (SDN): Layers and Architecture Terminology,” *hjp*. [Online]. Available: <https://www.hjp.at/(en)/doc/rfc/rfc7426.html>. [Accessed: 04-Apr-2021].
17. A. Kassler, L. Skorin-Kapov, O. Dobrijevic, M. Matijasevic, and P. Dely, “Towards QoE-driven Multimedia Service Negotiation and Path Optimization with Software Defined Networking,” *SoftCOM 2012, 20th International Conference on Software, Telecommunications and Computer Networks,* 2012. [Online]. Available: <https://ieeexplore.ieee.org/document/6347665?denied=>. [Accessed 25-Mar-2021].