Integrating Context Awareness and SDN for a Lightweight Approach to Adaptive Networking

José C. F. M. Júnior[2,1], David C. P. da Cunha[1], Carlos A. G. Ferraz[1]

[1]Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE) – Recife, PE – Brasil

[2]Engenharia e Operação de Redes, PoP-PE/RNP
Instituto de Tecnologia de Pernambuco (ITEP) – Recife, PE – Brasil

josecarlos@pop-pe.rnp.br, {dcpc,cagf}@cin.ufpe.br

Abstract. *With computer networks everywhere, managing them is becoming increasingly complex, leading to higher costs, longer response times, and more human errors in decision making. Given that, the search for a Smart Network Management, with less human intervention and more automation, becomes indispensable. Several technologies have been proposed to enable "Smart Management," including machine learning, notably for more complex cases. The Software-Defined Networking (SDN) paradigm is one of the most promising. Literature shows, however, that there are still several research challenges and opportunities for the automation of many network management tasks. This paper proposes to combine context awareness with SDN in a lightweight approach to Smart Management, or Adaptive Networking, to handle in a simplified way everyday events such as link-down and traffic congestion. A comparative analysis shows that the management approach combining context awareness with SDN is between 40% and 73% faster than the support of SDN to a human-driven management.*

1. Introduction

In the last decades, computer networks have become more and more ubiquitous, being used in an increasingly intense and diversified way. Advanced network services are essential to enable some of today's most varied and dominant topics such as cloud computing, the Internet of Things (IoT), complex streams processing, high-definition multimedia services, among many others. In this sense, network management requirements are increasingly complex and can lead to higher costs, higher response times and more human errors in decision making. These and other problems require less human intervention and more automation. Several technologies have emerged to enable self-management or "Smart Management", such as SDN (Software-Defined Networking) [Kim and Feamster 2013], integration of solutions based on machine learning [Ayoubi et al. 2018], and context-awareness [Bui et al. 2017]. However, according to [Ayoubi et al. 2018], the automation of many network management tasks has not yet been sufficiently explored, and therefore there are several open research opportunities and challenges. For [Rojas 2018], the Software-Defined Networking paradigm is still at the beginning of its development, and considering total automation and effortless management as the primary goal of these networks, a number of challenges are to be addressed, so there is still a long way to go for a fully autonomous network management vision [Khan et al. 2018].

In this paper, we propose to join the SDN paradigm with context-awareness to make network management more adaptive and, consequently, to reduce human intervention. Such a combination focuses on common network events, such as a 'link down' problem and network congestion, for example. SDN promises flexible, dynamic, programmable, and manufacturer-independent network management. The adoption of context-aware computing together with SDN to handle well-known events such as those mentioned above is efficient, in the sense of executing tasks usually performed by human resources in network management more quickly, transparently and with few or no errors, and yet simple, i.e., without the need to apply more sophisticated solutions based on machine learning, for example. This work defines four use cases by considering situations that commonly happen in networks. Three of them were emulated in a portion of ICONE (Infraestrutura de Comunicação Óptica para eNsino e pEsquisa, in Portuguese), the metropolitan education and research network of Recife, Brazil. The results of tests performed in the virtualized network show that the combined use of SDN technology with context-awareness, based on simple and intuitive rules or policies, allows the managed network to behave as expected, both adaptively and with satisfactory response times. This work helps to give back the network manager time to carry out other management activities, including designing new and improved advanced network services.

The remainder of this paper is organized as follows: next section (2) discusses Smart Management based on SDN, Machine Learning and Context-Awareness, as well as related works. Section 3 presents the proposal of the joint SDN-Context Awareness approach for Adaptive Network Management, followed by an implementation based on the ICONE network (section 4), where results are presented with a brief statistical analysis of them. Finally, conclusions and suggestions for future work are drawn in section 5.

2. Smart Management: Background and Related Works

In the area of computer network management, [Shu et al. 2016] ratify the general objective of maintaining availability and improving network performance, which has been pursued since traditional management, based on Network Management Systems (NMS), Simple Network Management Protocol (SNMP) and Management Information Base (MIB). Traditionally, network management requires actions that are heavily dependent on human administrators. To change this reality, research efforts have focused on agile and adaptive management architectures that support self-managing networks or *Smart Management*.

2.1. Smart Management and SDN

Reference [Ayoubi et al. 2018] follows this trend, placing Machine Learning (ML) as an option to provide cognitive management in SDN. Reference [Khan et al. 2018], on the other hand, considering the evolution of the mobile networks, conclude that the task of management dependent on human actions is increasingly complex. New dynamic adaptation requirements arise as a result of the increasing heterogeneity and complexity of mobile devices and applications, and the need for resilient networks, among others. According to [Lee et al. 2018], a "zero-touch network" [Koley 2016] is a network that minimizes service downtime and operating costs thanks to the removal of human intervention. It is built using cloud and SDN technologies [Van Rossem et al. 2017].

The SDN controller maintains a global view of the network and provides unified applications and policies. The instructions are provided by the controller, independently of vendors. Currently, the OpenFlow (OF) protocol [OPEN NETWORKING FOUNDATION 2015] is one of the most used for communication between the control and data planes, enabling SDN in network switches. The controller is the SDN network element that provides a programmable interface to the network, which enables the implementation of management tasks and new functionalities. Applications can be written in multiple languages, allowing interaction through a REST API [Nunes et al. 2014]. Examples of SDN controllers are OpenDaylight[1], Ryu[2] and ONOS[3].

2.2. Policy-Based Network Management, Machine Learning and SDN

In [Khan et al. 2018] the authors believed that Policy Based Network Management (PBNM) could be one of the key concepts towards self-x[4] NM. The motivation is to offer the network providers a set of abstractions for dealing with the associated complexity, decreasing the gap between business and NM levels. The ideia is to combine top-level networking policies and low-level network configuration [Shirmarz and Ghaffari 2020]. Policies separate the rules that govern the system's behavior from their functionality. They present a simple way to decouple the implementation and behavior of a network entity by allowing the manipulation of operational constraints during runtime without interfering with the source code. Reference [Hadjiantonis 2012] mentions, however, that the complexity of realizing policy-based NM solutions in practice limits self-management capabilities in large-scale networks.

Moving on to network management (NM) based on Machine Learning (ML), [Ayoubi et al. 2018] remarked that ML is a powerful technique for extracting knowledge from data, but its potential is still seldomly used for practical solutions in adaptative networking. However, with the programmability of SDN, the large amounts of current data sources and the high availability of computing power delivered by cloud computing, it becomes feasible the emergence of ML-based NM. Reference [Liu and Xu 2019] describes the use of ML techniques combined with the paradigm of SDN. The separation of the data and control planes allows the administrator to manage and control the network through programming techniques, which allows incorporating machine learning. The ML techniques involve some tasks, such as collection and training (algorithms) to classify data, and security issues, increasing the level of computational complexity.

2.3. Context Awareness

Context awareness would be the ability of the computer system to observe characteristics and identify changes in the environment around it and, in addition, be able to react to such changes. Therefore, current context-aware systems operate in three stages: (1) **Data Collection**, which can occur via sensors, historical system information, or other applications; (2) in the **Context Inference** (or *reasoning*) stage, rules are defined and the system interprets the occurrence of different situations of interest. In general, an approach based on the static and prior definition of rules by administrators can be used, or based on

[1]https://www.opendaylight.org/
[2]https://ryu-sdn.org/
[3]http://www.onosproject.org/
[4]Self-configuration, Self-healing, etc.

automatic decisions using machine learning algorithms, for example; (3) in the **Context Adaptation** step, the system can make decisions and perform customized actions.

Contrary to what one might imagine, Context Awareness and Machine Learning are not rival technologies, but rather complementary, just like Context-Aware and Policy-Based Network Management can be combined. In fact, they can be used separately in various application domains that aim to minimize the role of the human in the control loop [Ayoubi et al. 2018, Bui et al. 2017], but also together [Bui et al. 2017], with improved results, in general [e Silva 2016].

This paper focuses on the use of context awareness based on simple rules or policies in network management. These rules are considered simple because they are based on the network manager's intuition (knowledge, experience) [Rojas 2018] – eg., if data throughput is greater than 80% of the bandwidth, then the network is considered *congested*. Simple context-awareness techniques, such as key-value pair, are used in everyday events of today's networks (eg., link down, network congestion), which in general do not justify the use of more complex techniques, such as machine learning.

3. SDN and Context for Adaptive Network Management

This work proposes the combined use of SDN and context awareness, based on simple rules or policies for *adaptive network management*. The following describes the implementation, based the software-defined networking (SDN) paradigm, of the steps for achieving context awareness (**data capture**, **context inference**, and **adaptation**).

3.1. Data capture

Capturing data involves sensing and collecting specific stored information that can define a particular situation. To identify a *link down*, we capture an event related to a port status. We implemented in Python the decorator `@set_ev_cls()` defined by the `ryu.controller.handler` module, pointing to the `ryu.controller.dpset.EventPortModify` class. In addition, to detect a *heavy congestion* event, we implemented the traffic monitor described in the Ryu documentation. At every 1 second, we make an analysis of the contextual element *throughput* in Mbps, then we perform a simple bandwidth calculation on each port of the emulated network switches as seen in Listing 1 [Megyesi et al. 2017] [Shu et al. 2016].

```
1  throughput = (Byte_fin - Byte_ini)*8
2  result = throughput/1048576
```

Listing 1. Capture Contextual Information (Throughput)

3.2. Context Inference and Adaptation

This stage treats the information regarding the events *link down* and *heavy congestion*. These data are processed by rules that are described below. After a *link down* event is captured, the inference stage analyzes the link state of the switch port through the `EventPortModify` class (Listing 2). The `EventPortModify` class of the Ryu controller has an attribute structure in Python that returns information about the link. We can

access this attribute through `ev.msg`. We process this message to retrieve information about the link. If the Link is `DOWN`, the adaptation process involves the modification of the traffic direction to the port that was not affected by the event.

```
1  if ev_list[2] == 'DOWN':
```

Listing 2. Context Inference Link Down

The rule for a *heavy congestion* inference, which checks if $throughput \geq 80\%$ of the link bandwidth, is seen in Listing 3 - note that `throughput` (`result`) is calculated as in Listing 1.

```
1      if (result >= MAX_BAND*0.8):
```

Listing 3. Context inference for Heavy Congestion

The adaptation approach involves the modification of the flow entry of the datapath. On the statement above, we show that, if the $result$ (throughput) is equal or greater than 80% of the maximum bandwidth, we will change the traffic direction. The change implies in deleting and adding new flow entries on the affected switches. We could achieve this defining 2 simple rules (Listing 4), where `out_ports` represents the port affected by the *link down* or the *heavy congestion*, which shows: if the affected port (out_ports) is 3, change the traffic to port 2, or if the affected port is 2, change the traffic to port 3.

```
1  if out_ports == 3: out_ports = out_ports - 1
2  elif out_ports == 2: out_ports = out_ports + 1
3  else: pass
```

Listing 4. Context Adaptation for Link Down and Heavy Congestion

4. Experimental Methodology and Analysis

We defined the **methodology** for the execution of the project based on the use of emulation and experimentation in the field. **Emulation**, in a software context refers to the use of an application or tool to mimic the behaviour of other software or device (hardware). In this work, *Mininet* (http://mininet.org/) has been used to emulate the behavior of a part of the ICONE[5] network (Figure 1) for management purposes associated with SDN technology, using the Ryu controller. In a next step, **controlled experiments** should be carried out in the ICONE network itself, to verify the adequacy of the combined use of SDN with context in network management (verification of the *cause-effect* relation).

Besides the PoP-PE node (the Point of Presence in Pernambuco of the National Research and Education Network – RNP) that serves as the connection point of the ICONE Network with RNP/Internet, the selection criteria for the other three nodes of the virtualized network (a small portion of ICONE) were:

[5]ICONE stands for **I**nfraestrutura de **C**omunicacão **Ó**ptica para e**N**sino e p**E**squisa.

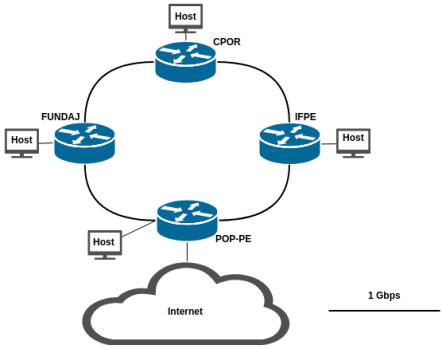

Figure 1. Emulated portion of the ICONE network.

- CPOR (Centro de Preparação de Oficiais da Reserva – Brazilian Army): a small traffic generator, located halfway to PoP-PE in both directions on the ring network;
- FUNDAJ (Fundação Joaquim Nabuco/CGF – Gilberto Freyre Campus): potentially large traffic generator, located to the west of both PoP-PE and CPOR;
- IFPE (Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco): frequent large traffic generator, located to the east of PoP-PE and CPOR.

4.1. Architecture

The system architecture is shown in Figure 2. Initially, an emulation and analysis environment was created to emulate a virtual network using the OpenFlow protocol and the Software-Defined Networking (SDN) approach to manage network behavior dynamically (aware of context).

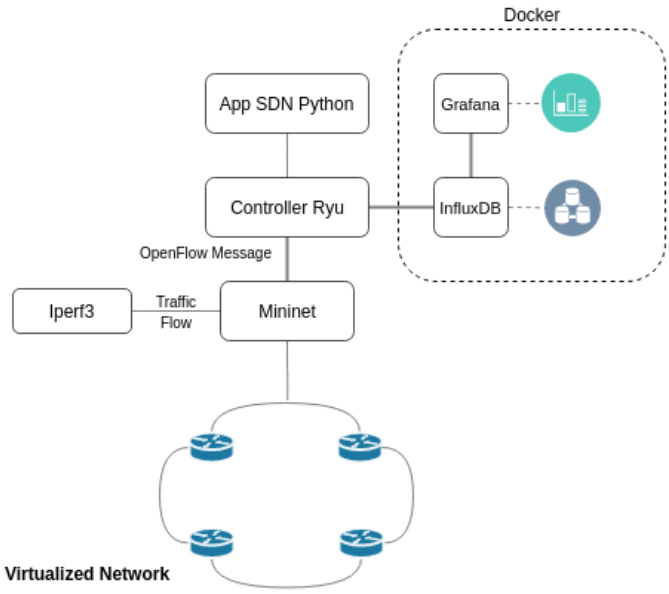

Figure 2. System Architecture.

A *Context-Aware Network Management Application*, implemented in Python, receives, through the *Northbound API* of the Ryu Controller, data related to the traffic of the virtualized network, infers context (e.g., "Heavy congestion on IFPE → POP-PE link")

and adapts to such a context (e.g., redirect traffic to an alternative link), enabling the adaptation of the performance of the network. This way, it contributes to reduce the human intervention in the management activities of the network.

Network performance testing requires a multi-rate traffic generation and measurements of network throughput, packet loss, latency, jitter, and so on. *IPerf3* (http://iperf.fr/) was the tool chosen to perform network throughput measurements in the experiments. For the purpose of visual analyses and alerts, we used *Grafana* (https://grafana.com/) for dealing with graphics. Besides that, *R* (http://www.r-project.org/) was used for statistical analyzes.

4.2. Use cases

To test the combined use of context-awareness and SDN in the ICONE network management, we have designed four scenarios that commonly occur: (1) "Normal" traffic situation; (2) "Link-down" situation; (3) "Heavy" traffic/congestion situation; (4) "Moderate" traffic/congestion situation.

Intuitively, we have configured the *traffic parameters* as exemplified in Figure 3. Since the "Moderate" traffic situation (4) has not yet been implemented, it is not going to be dealt with in this article. Also, regarding the 'Normal" traffic situation (1), which is observed while the $throughput \leq 60\% \; of \; the \; bandwidth$ (Figure 3), the management application simply sets the FLAG `traffic_context` to `NORMAL` and goes on running without the need to adapt. Thus, only the scenarios (2) and (3) are discussed in the following.

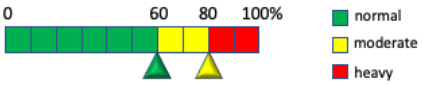

Figure 3. Example of traffic parameters configuration. Based on intuition, a network manager can configure them at will.

4.2.1. Link-down

In this case, the SDN controller captures the event when the link between two switches is unavailable (down). In (1) the rule that infers the context simply checks the state of the link and, if it is `DOWN`, immediately adapts to it by changing (inverting) the flow in the ring network that emulates ICONE.

$$\textbf{If } Link\_State = Link\_Down \textbf{ then } Change\ Flow \tag{1}$$

We emulated the link-down scenario according to the following steps:

1. a host (h3) in CPOR was prepared to send data, initially, in the CPOR $\rightarrow$ IFPE $\rightarrow$ POP-PE direction with throughput of $\approx$200 Mbps – see Figure 4(`a-b`) (*Inbound interface s1_POP-eth3*);
2. a host (h4) in IFPE was prepared to send data $\approx$400 Mbps in the IFPE $\rightarrow$ POP-PE direction (output $\approx$ 200+400 Mbps – Figure 4(`b-c`));

Figure 4. Inbound traffic changes due to link-down.

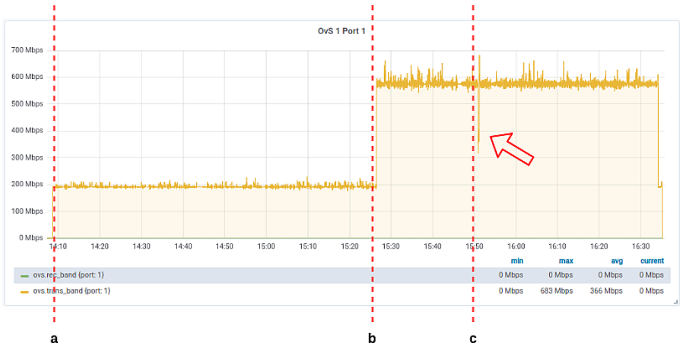

Figure 5. Outbound traffic changes due to link-down.

3. the CPOR ↔ IFPE link was dropped – Figure 4(c) to represent the "link-down" situation;
4. quickly (see response times in section 4.3) after time c, the flow coming out from CPOR changes the route, moving the traffic to the CPOR → FUNDAJ → POP-PE direction (interface s1_POP-eth2 – *Inbound*).

In the graphics (Figure 5) that shows the interface `s1_POP-eth1` (*Outbound*), it is possible to observe (highlight arrow) a drop in the data output at time c, and right after that the traffic returns to normal (step (4) above).

4.2.2. Heavy congestion

In this case, the traffic practically does not flow as it is heavily congested, leading to the need for a route change. The rule defined to infer this scenario is as follows.

$$\textbf{If } Throughput \geq 80\% \; of \; the \; bandwidth \textbf{ then } traffic\_context = Heavy \tag{2}$$

Here the inference rule has its contextual analysis based on the statistical information of the throughput and bandwidth. In this scenario, it is inferred that the traffic is very congested when $throughput \geq 0,8 \times bandwidth$ (e.g., if the available bandwidth on a given link is equal to 1 Gbps, the congestion is considered "heavy" if its throughput is greater than or equal to 800 Mbps).

4.3. Evaluation

To verify the efficiency of this proposal, we performed two experimental statistical analyses. In the *first experiment* we collected samples regarding scenario *heavy congestion* with size greater than 30 (repetitions) based on the Central Limit Theorem as described by [Larson and Farber 2010, p. 221]. Such samples refer to the capture of data relating to the Response Time (RT) at the steps: (a) Capture – collect network metrics (RTa), (b) Inference – apply specific rule and become aware of the context (RTb), and (c) Adaptation – adapt to the context (RTc). In the descriptive analysis of the Response Times (RTa, RTb, RTc), median values were 8.48, 6.90, 1.78 milliseconds (with a standard deviation of 2.18, 1.92, 0.56 ms), respectively. This time difference is justified because step (a) takes longer to calculate the throughput in real time based on Openflow flows, while (b) takes a little less time to be aware of the context and infer a congestion situation, and adaptation (c) is a simpler process in which output ports are inverted (Listing 4).

In the *second analysis*, we compare the approaches called SDNC – experiments conducted in an SDN environment with context-based network management, and SDNH – experiments conducted in an SDN environment supporting a traditional human-driven management. Comparing the two approaches regarding the descriptive statistics analyses of the two samples (SDNC and SDNH), we observed that in the SDNC approach, which involves only the Capture (a) and Inference (b) steps, the median of the sampled values equals to 15.007 ms, slightly shorter than the median of the sampled values in the SDNH approach, as for a human manager a step (d) of generating a display alarm is needed, so it slightly increases the response time to 15.082 ms. Considering [Nicolaou 1990], which indicates that the *human perception threshold time* in user interfaces (UI) should be on the order of 10–40 ms (note the additional sequence of steps: (d) Alarm – alarm generation to appear on the management UI (RTd), and (e) Human Perception of the alarm (RTe), to come before the Adaptation (c) step), in the SDNH approach a response time is between 25.082 ms to 55.082 ms, much higher than the 15.007 ms of the SDNC approach, as $SDNC = RTa + RTb > SDNH = RTa + RTb + RTd + RTe$ (between 40% and 73% faster). Therefore, our lightweight (context-aware) approach to adaptive network management can be considered to be feasible. We compared SDNC and SDNH without RTc, as the adaptation process would be significantly different in both approaches. The previous analysis showed that RTc is very short (1.78 ms) in SDNC, whereas for SDNH, one naturally expects that the human adaptation to a network situation is on the order of a few seconds.

5. Conclusions

This work took advantage of the software-defined networking paradigm (SDN) to implement a context-aware approach to *adaptive networking*. Context awareness uses relatively simple and intuitive rules to infer a situation (e.g., heavy congestion) and make the network automatically adapt to the established context. Context-awareness comprises three

stages: (1) data capture, (2) context inference, and (3) adaptation to the inferred context. The results shown in this paper, regarding the response times of the system for sensing the network and inferring the context (stages 1 and 2), show that in common scenarios, such as *link down* and *traffic congestion*, the use of context-awareness is sufficiently adequate and less complex than using machine learning, for example. Considering total adaptation and effortless management as the primary goal of software-defined networking, this work gives a simple but important contribution to the vision of fully autonomous network management. It helps to give back the network manager time to carry out other activities, including designing new and improved advanced network services.

References

Ayoubi, S., Limam, N., Salahuddin, M. A., Shahriar, N., Boutaba, R., Estrada-Solano, F., and Caicedo, O. M. (2018). Machine learning for cognitive network management. *IEEE Communications Magazine*, 56(1):158–165.

Bui, N., Cesana, M., Hosseini, S. A., Liao, Q., Malanchini, I., and Widmer, J. (2017). A survey of anticipatory mobile networking: Context-based classification, prediction methodologies, and optimization techniques. *IEEE Communications Surveys Tutorials*, 19(3):1790–1821.

e Silva, D. V. (2016). *CD-CARS: Cross-Domain Context-Aware Recommender Systems*. PhD thesis, Universidade Federal de Pernambuco. Ciências da Computação.

Hadjiantonis, A. M. (2012). Autonomic management of mobile and wireless networks. In *Telecommunication Economics*, pages 199–208. Springer, Berlin, Heidelberg.

Khan, M. A., Peters, S., Sahinel, D., Pozo-Pardo, F. D., and Dang, X.-T. (2018). Understanding autonomic network management: A look into the past, a solution for the future. *Computer Communications*, 122:93 – 117.

Kim, H. and Feamster, N. (2013). Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119.

Koley, B. (2016). The zero touch network. *2016 IEEE 12th International Conference on Network and Service Management (CNSM)*.

Larson, R. and Farber, B. (2010). *Estatística Aplicada*. Pearson Prentice Hall, São Paulo, Brasil, 4 edition.

Lee, Y., Vilalta, R., Casellas, R., Martínez, R., and Muñoz, R. (2018). Auto-scaling mechanism in the ict converged cross stratum orchestration architecture for zero-touch service and network management. In *2018 20th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4.

Liu, J. and Xu, Q. (2019). Machine Learning in Software Defined Network. *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, (Itnec):1114–1120.

Megyesi, P., Botta, A., Aceto, G., Pescapé, A., and Molnár, S. (2017). Challenges and solution for measuring available bandwidth in software defined networks. *Computer Communications*, 99:48–61.

Nicolaou, C. (1990). An architecture for real-time multimedia communication systems. *IEEE Journal on Selected Areas in Communications*, 8(3):391–400.

Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., and Turletti, T. (2014). A survey of software-defined networking. past, present, and future of programmable networks. *IEEE Communications Surveys Tutorials*, 16(3):1617–1634.

OPEN NETWORKING FOUNDATION (2015). OpenFlow Switch Specification 1.5.1.

Rojas, E. (2018). From software-defined to human-defined networking: Challenges and opportunities. *IEEE Network*, 32(1):179–185.

Shirmarz, A. and Ghaffari, A. (2020). Performance issues and solutions in sdn-based data center: a survey. *The Journal of Supercomputing*, 76(10):7545–7593.

Shu, Z., Wan, J., Lin, J., Wang, S., Li, D., Rho, S., and Yang, C. (2016). Traffic engineering in software-defined networking: Measurement and management. *IEEE Access*, 4:3246–3256.

Van Rossem, S., Cai, X., Cerratoz, I., Danielsson, P., Németh, F., Pechenot, B., Pelle, I., Risso, F., Sharma, S., Sköldström, P., and John, W. (2017). NFV service dynamicity with a DevOps approach. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 865–866.