FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# API design and implementation of management interface for SDN whitebox switches

**Rubens Jesus Alves Figueiredo**

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Ana Cristina Costa Aguiar

December 21, 2017

# Resumo

# Abstract

...

# Contents

# List of Figures

# List of Tables

# Abreviaturas e Símbolos

| | |
|---|---|
| SDN | Software Defined Networks |

# Introduction

## 1.1 Context

## 1.2 Motivation

## 1.3 Goals

# Berlin Institute for Software Defined networks

# Software Defined Networking

Computer networking is a vital part of the services that are offered today, and as such, the performance in technology backing these services is central to the quality of these services. As the service providers reorganize their data centers in the cloud computing domain, enabling several improvements in the predictability, quality of service and ease of use of their services. New technologies are then required to make sure that their services are adapted to the fast changing landscape of networking services. One of the most notable innovations in this field is called Software Defined Networking, because its architecture allows for two essential features

- **Separation of network planes** SDN allows for the separation of the network control plane from the data forwarding plane by having network "intelligence" present in the network controllers, and having them control the forwarding elements that live in the Data Plane
- **Centralization of network management functions** By isolating the management on a separate plane, there is possibility of developing a single controller that can regulate the entire network, having unrestricted access to every element present in the network, simplifying management, monitoring, application of QoS policies, flow optimization, ...

In this chapter we explore the essential characteristics of SDN, the technologies that provide the back end for the development of this technology, and current implementations of the most popular SDN controllers, so that we can see the features that should be present while developing a management interface for SDN controllers.

## 3.1 OpenStack

## 3.2 OpenFlow

As the growth of the networking infrastructure of the past few decades became evident, the need for an enviroment that allows for experimentation and testing of different protocols and equipment became evident. If networking research would depend on the previously existing methods, then new ways of creating and developing protocols would become increasingly hard to implement and develop. As such, there was need for a framework that could enable testing of new ideas on close to realistic settings. So, on 28 February 2011 the version 1.1 of OpenFlow was released, and this proposal quickly became the standard for networking in a Software Defined Network. Since 2011, this protocol has suffered some revisions, and the latest version supported is version 1.5.1. Since this framework has evolved quite a bit, this section focuses on the versions 1.3, which are the versions that are used in development of this dissertation.

Several reasons led to the quick standardization of this protocol, which are related not only to the initial requirements of the platform, like the capability of supporting high-performance and low-cost implementations, and the capability of ensuring separation between production and testing traffic, but also the extensibility that the open source development model provides, removing the limitations that closed or commercial solutions give the network researchers.

The big advantage of OpenFlow is that it is, from the data forwarding plane point of view, easy to process. Since the control decisions are made by the controller, which lives in a separate plane, all the switch needs to do is correctly match the incoming packets, and forward them according to the rules established by the controller. The components that are part of this system and enable this functionality are:

- **FlowTables** This element describes the main component of the switching capabilities of the OpenFlow switch. Inside the switch there are several flow tables that can be used to match incoming packets, and process them in the rules that are specified by the controller. These rules can contain actions that affect the path of the packets, and these actions usually include forwarding to a port, packet modification, among others. Classification is done via matching one or more field present in the packet, for example the switch input port, the MAC and IP addresses, IP protocol, basically all information required to correctly process the incoming packet. The required actions for an OpenFlow switch are the capability of forwarding to a set of output ports, allowing the packet to move across the network; to send them to the controller, in the case of a miss of match; and finally the ability to drop packets, which is useful for DDoS mitigation, or more security concerns.

- **OpenFlow Protocol** Through the establishment of the OpenFlow Protocol between the switch and the controller, there is the definintion of several messages that allow for the control of the switch. This protocol enables capabilities such as adding, deleting and updating flow mods in the switch, that are refered to as *Controller-to-Switch* messages. Other relevant message types are the *Asynchronous*, that enable the notification of some event that ocurred, this type includes the Packet-In message, that is a type of message that is sent to the controller when a certain packet has no match in the flow tables present in the switch; and the *Synchronous* message that enable functionality such as the Hello message, that is used to start the connection between the switch and the controller.

- **Secure Channel** OpenFlow defines the channel that is between the switch and the controller as a secure communications channel. As the messages that are sent to the switch are critical for the correct operation of the system, as indicated in the previous point, the channel should be criptographically secure, to prevent spoofing of this information. As such, the channel is tipically transported over TLS.

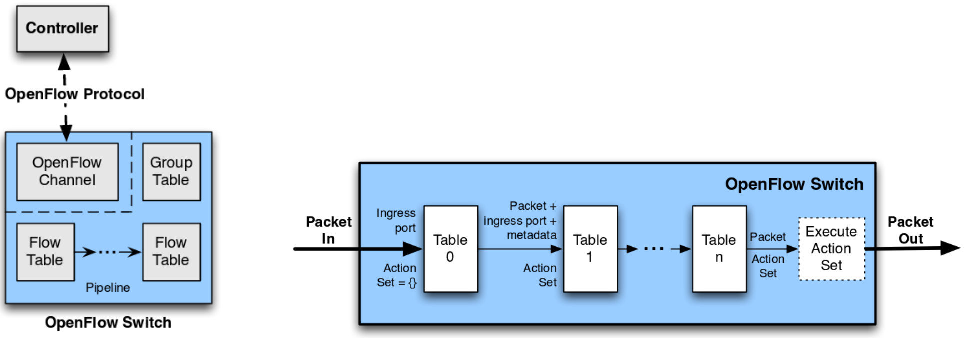


Figure 3.1: Images describing OpenFlow components. On the left, an overview to the entire system, and on the right a view at the table structure of the OpenFlow Switch

### 3.2.1 Open vSwitch

Although the OpenFlow protocol itself is well developed and well documented, there still needs to be support from switch applications to support this protocol. Although hardware switches are usually very expensive, and

#### 3.2.1.1 Open vSwitch

## 3.3 SDN Controllers

### 3.3.1 Floodlight

### 3.3.2 OpenDaylight

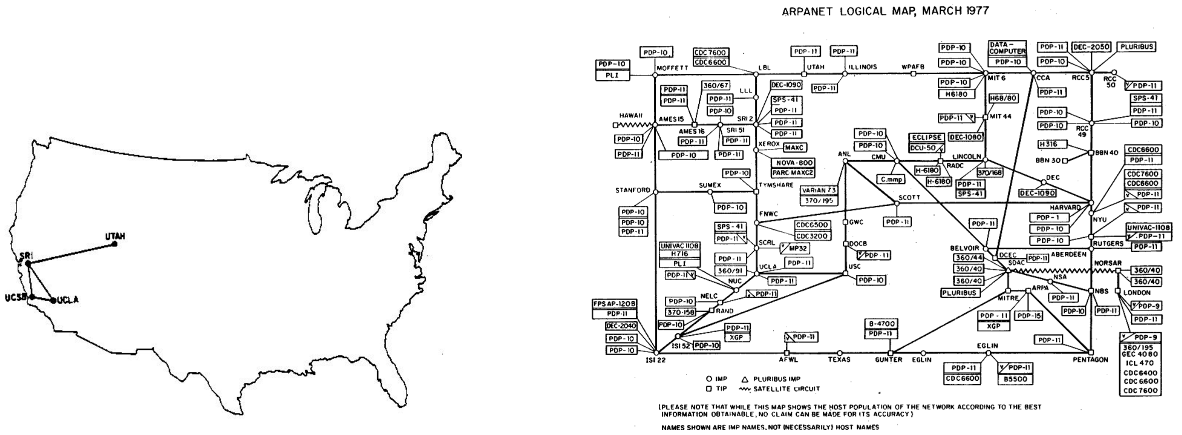
## 3.4 SDN Northbound

# Literature Review

## 4.1 Computer Networking

### 4.1.1 Historical context

A computer network is a way to transfer digital information from point A to point B, via an established link between the two. In the early days, the demand to create an interconnected network of data sharing appeared from academic research and military needs, and since the introduction of these innovations, many American universities started to join in the this network, called ARPANET.



(a) Early ARPANET schematics, appr. 1969

(b) More sites connected to the ARPANET, September 1977

Figure 4.1: ARPANET evolution

As the advantages of having an interconnected network of computers became clearer, and with the surge of some others, such as CYCLADES, the french investigation research network, the need to connect the existing networks was rising, and that was one of the first steps of creating a global network, later known as the Internet. Some of the essential mechanisms that can still be found to this day were also developed in the ARPANET, like FTP and e-mail.

One of them was introduced in 1981, RFC 793 [?], and with it TCP was "invented". The main motivation for this development was the introduction of an end-to-end, connection oriented, and reliable protocol that allowed for the standardization of several different protocols. Also in this document, the definition of a OSI model, like the one that is prevalent today, or the definitions of reliability, are present, and continue to be relevant until today.

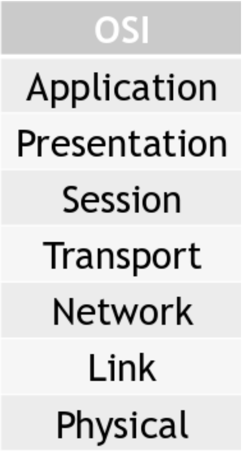


Figure 4.2: The current OSI model

One

### 4.1.2 Market data

By continuing to evolve and increase in both functionalities and users, the Internet as we know it is a global network, encompassing several protocols, and allows for instant communication of people around the world. A report indicating this evolution allows for some interesting conclusions about the state of the Internet market until 2021. This forecast was developed based on data originating from projections made from some Telecom and Media groups, direct data collection, and some estimates.

**Global IP traffic** As the report mentions, the monthly traffic, per capita, in 2016 is around 13 GB, and in 2021 is projected to be at 35 GB

**Mobile devices traffic** While today wired devices still make up for the majority of IP traffic, in 2021, traffic originating from Wi-Fi and mobile devices should account for 63 percent of the total traffic

**Smartphone/ PC traffic** By comparing the predicted evolution of smartphone/ pc traffic, the trends indicate that smartphone traffic should exceed fixed PC traffic.

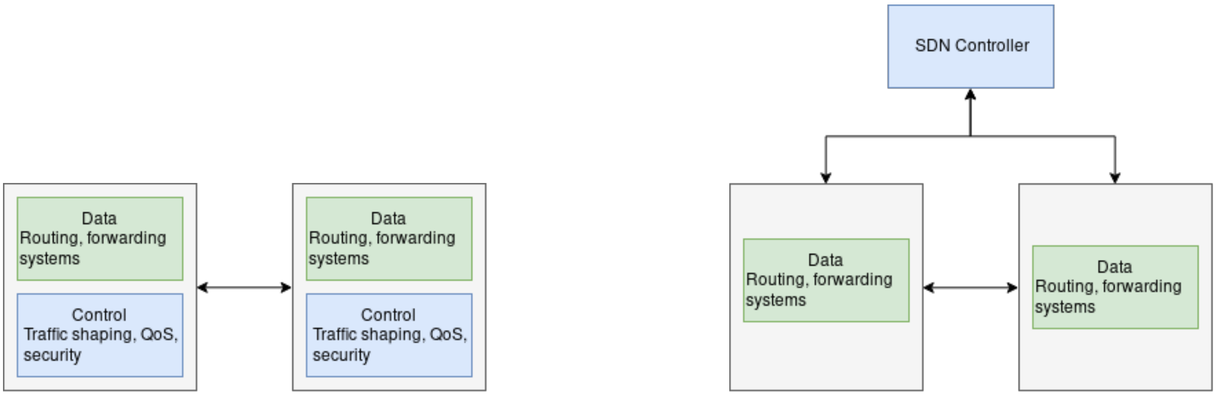
The previous points while obvious estimations, show a clear evolution in the way that Internet is usually accessed, and that is the

## 4.2 Software Defined Networking

As described in the previous section, there is a clear evolution of requirements, and this evolution was possible due to the adaptation of the exiting technologies to support better, and more efficient protocols that could carry the large amount of data that is transmitted every second. With that in mind, and in order to reduce costs to the service providers, simplify deployment and maintenance operations, developments in Software-Defined Networking (SDN) and Network Function Virtualization have been growing since 2010.

This new paradigm introduces programmability in the configuration and management of networks, by consolidating the control of network devices to a single central controller, achieving separation of the control and the data plane, and supporting a more dynamic and flexible infrastructure. Another important paradigm, that follows the development of SDN, is the concept of Network Function Virtualization. This concept allows to remove the amount of *middleboxes*[1], by replacing these with generic software applications.

The essence of SDN/NFV is described in a short manner if the Open Networking Foundation (ONF) paper: *In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications*. The following picture defines both of the approaches to network, the traditional one, where the data and control plane are just one, and the SDN way, that considers that application and control traffic should be considered in two different ways.


Data
Routing, forwarding systems
Control
Traffic shaping, QoS, security
Data
Routing, forwarding systems
Control
Traffic shaping, QoS, security
SDN Controller
Data
Routing, forwarding systems
Data
Routing, forwarding systems

(a) Traditional networking architecture

(b) SDN architecture

Figure 4.3: Traditional vs SDN network architecture

[1] Computer networking device that does some operations on traffic, excepting packet forwarding. Examples include caches, IDS's, NAT's, ..

One example of the operation of a switch in the SDN model is the following:

- A switch runs an agent, and this agent is connected to a controller;
- This controller runs software that can operate the network, managing flow control rules, and collecting information, enabling a full view of the network from a central node
- While this controller can be logically centralized, for fault-tolerance and high availability purposes it can be distributed, and by optimizing datamodels and providing caches, one of the earliest ONOS prototypes was able to achieve a distributed Network OS that could be applied to production networks [12]

More details on the technology that runs this model can be found in the next chapter.

After analysing some examples of deployment of the SDN model, the analysed literature provides some insights on the requirements that platforms using this paradigm need to obey [3].

- **High Performance**
- **High Availability**
- **Fault Tolerance**
- **Monitoring**
- **Programmability**
- **Security**

### 4.2.1 Why?

### 4.2.2 Challenges

### 4.2.3 Observed implementations in the industry

## 4.3 Cloud Computing

## 4.4 Databases

# Network Management

As networks grow larger and more complex, systems must be put in place that allow for closely monitoring the resources that make up the network, while also allowing for a certain freedom for the possible constant change of the network. As such, typical vendor solutions don't really fit into this ever changing landscape, since they present very solid and vertically integrated solutions. The SDN paradigm, however, is able to solve this issue, since it enables for the centralized control of the underlying networks, which provides visibility and even control over the network, simplifying network diagnosis or troubleshooting.

Although SDN is a promising paradigm in terms of networking management, it also introduces some points of failure that are non existing, or not as impactful in current networking deployments. This is related, for example, to the centralization of the controller, which makes it susceptible to Denial-of-Service attacks or even the possibility of some malicious attacker that could possibly exploit the privileged view that the SDN controller has.

This chapter focuses on the management of networks, where we explore what is most necessary to obtain a comprehensive understanding of the network; formalize the statistics that can be reported via OpenFlow; see some research that has been done in the management of SDN applications, including what existing controllers provide us; and finally explore the way that DDoS detection and mitigation is usually implemented.

The topic of network management is very extensive, due to the several components that make up today's networks, and the vast amount of information that they provide. It can be summed up as the operation and maintenance of network infrastructure so that the service it provides is not only "healthy", but also is operated at a level that keeps costs down for service providers. While obviously important for all network operators, these are some very important concerns that should be taken care in cloud service data center networks. The modern cloud computing centers are subject to several different characteristics than more traditional infrastructures, due to the increasingly predominance of virtualisation of server and networks or the pretty specific topology types that can be present in these deployments, as can be seen in figure 4.1. Other possible topologies include de Bruijn server only networks, or BCube switch heavy networks [**?**].

A network management system usually consists of a centralized station, and management agents running on the network devices. Using management protocols, the agents can report to the station infor-

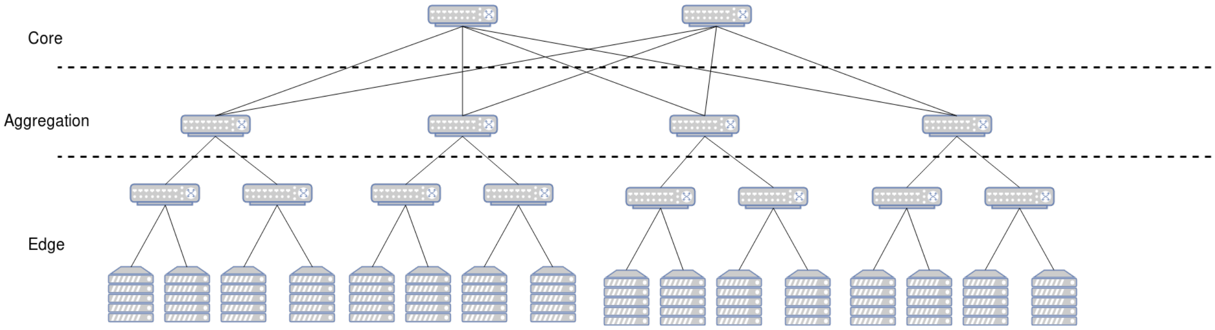


Figure 5.1: Visual representation of the fat tree topology commonly used in data centers

mation about the its operational status, which includes information ranging from CPU load to bandwith usage. Typically this information can be retrieved by the controller polling the agents, or the agents sending information on their own, usually to inform status changes. Using this information, the network operator can get insight on the performance or possible errors of the devices that are monitored. In the next section, we explore one of the most popular management protocols, SNMP.

## 5.1 SNMP

The Simple Network Management Protocol is an IETF defined protocol that allows for the interconnection of networking devices, and provide a structured way to retrieve relevant information about these devices. As the name suggests, SNMP allows for a simplified approach to network monitoring, since it reduces the complexity of the functions that the management agent needs to comply with, which bring several advantages, like reducing the costs for development of management tools; provides a way to monitor, independently from different hardware providers the resources; and also supports freedom in extending the protocol in order to include other aspects of network operation. [**?**]

The architectural model of SNMP can be described as the following components:

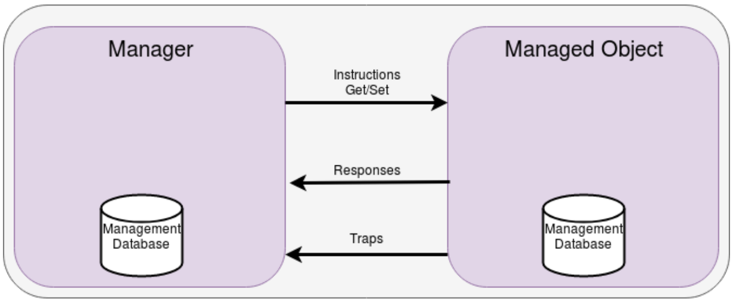
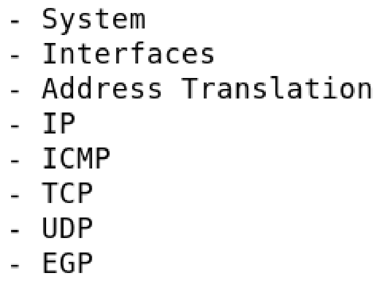


Figure 5.2: Architectural components of SNMP

The management database is one of the most important components of this system, because it serves as a reference to the entities that are managed in the SNMP protocol. The formal name for this database is the MIB - Management Information Base [?], and its composed of a collection of the objects and their variables. Figure 5.4 features the object groups that make up the MIB.



```
- System
- Interfaces
- Address Translation
- IP
- ICMP
- TCP
- UDP
- EGP
```

Figure 5.3: Object groups that compose the MIB [?]

The MIB is structured in a tree-like hierarchy. This structure allows for the organization of all objects in a logical pattern, as there is a parent node, that does not have any label, but provides three children, that provide different indexes for different organizations, so that each can maintain their own standards. One advantage of this system is that companies can maintain their own MIB, while also remaining compatible with the rest of the standard.

Due to its permanence in the market, the protocol has suffered some large changes since its original design. SNMPv3 now supports important changes to the original one, most notably in the security aspects, introducing strong authentication and encryption capabilities.

## 5.2 OpenFlow

Looking now at OpenFlow related monitoring, some of the

# API Design

# Existing System

# References

[1] M. Schwartz and N. Abramson. The Alohanet-surfing for wireless data [History of Communications]. *IEEE Communications Magazine*, 47(12), 2009.

[2] Ian F. Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou. A roadmap for traffic engineering in SDN-OpenFlow networks. *Computer Networks*, 71:1–30, October 2014.

[3] Hitoshi Masutani, Yoshihiro Nakajima, Takeshi Kinoshita, Tomoya Hibi, Hirokazu Takahashi, Kazuaki Obana, Katsuhiro Shimano, and Masaki Fukui. Requirements and design of flexible NFV network infrastructure node leveraging SDN/OpenFlow. In *Optical Network Design and Modeling, 2014 International Conference on*, pages 258–263. IEEE, 2014.

[4] Hiroaki Hata. A study of requirements for SDN switch platform. In *Intelligent Signal Processing and Communications Systems (ISPACS), 2013 International Symposium on*, pages 79–84. IEEE, 2013.

[5] Sakir Sezer, Sandra Scott-Hayward, Pushpinder Kaur Chouhan, Barbara Fraser, David Lake, Jim Finnegan, Niel Viljoen, Marc Miller, and Navneet Rao. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7):36–43, 2013.

[6] Amin Tootoonchian, Sergey Gorbunov, Yashar Ganjali, Martin Casado, and Rob Sherwood. On Controller Performance in Software-Defined Networks. *Hot-ICE*, 12:1–6, 2012.

[7] Amin Tootoonchian, Sergey Gorbunov, Yashar Ganjali, Martin Casado, and Rob Sherwood. On Controller Performance in Software-Defined Networks. *Hot-ICE*, 12:1–6, 2012.

[8] Md. Faizul Bari, Raouf Boutaba, Rafael Esteves, Lisandro Zambenedetti Granville, Maxim Podlesny, Md Golam Rabbani, Qi Zhang, and Mohamed Faten Zhani. Data Center Network Virtualization: A Survey. *IEEE Communications Surveys & Tutorials*, 15(2):909–928, 2013.

[9] Christos Douligeris and Aikaterini Mitrokotsa. DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44(5):643–666, April 2004.

[10] Hyojoon Kim and Nick Feamster. Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119, 2013.

[11] Keith Kirkpatrick. Software-defined networking. *Communications of the ACM*, 56(9):16, September 2013.

[12] Pankaj Berde, William Snow, Guru Parulkar, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, and Pavlin Radoslavov. ONOS: towards an open, distributed SDN OS. pages 1–6. ACM Press, 2014.

[13] Jan Medved, Robert Varga, Anton Tkacik, and Ken Gray. Opendaylight: Towards a model-driven sdn controller architecture. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, pages 1–6. IEEE, 2014.

[14] Robert W. Merriam and Elisabeth Feil. The potential impact of an introduced shrub on native plant diversity and forest regeneration. *Biological Invasions*, 4(4):369–373, 2002.

[15] Abraham Yaar, Adrian Perrig, and Dawn Song. Pi: A path identification mechanism to defend against DDoS attacks. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 93–107. IEEE, 2003.

[16] Jurgen Schonwalder, Martin Bjorklund, and Phil Shafer. Network configuration management using NETCONF and YANG. *IEEE communications magazine*, 48(9), 2010.

[17] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, and Darrell Kindred. Statistical approaches to DDoS attack detection and response. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 303–314. IEEE, 2003.

[18] Marco Canini, Dejan Kostic, Jennifer Rexford, and Daniele Venzano. Automating the testing of OpenFlow applications. In *Proceedings of the 1st International Workshop on Rigorous Protocol Engineering (WRiPE)*, 2011.

[19] Lucian Popa, Sylvia Ratnasamy, Gianluca Iannaccone, Arvind Krishnamurthy, and Ion Stoica. A Cost Comparison of Datacenter Network Architectures. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 16:1–16:12, New York, NY, USA, 2010. ACM.