On Designing SDN Services for Energy-Aware Traffic Engineering

Marcos Dias de Assunção(✉), Radu Carpa, Laurent Lefèvre, and Olivier Glück

Inria Avalon, LIP Laboratory, École Normale Supérieure de Lyon,
University of Lyon, Lyon, France
{marcos.dias.de.assuncao,radu.carpa,laurent.lefevre,
olivier.gluck}@ens-lyon.fr

Abstract. As experimenting with energy-aware techniques on large-scale production infrastructure is prohibitive, several traffic-engineering strategies have been evaluated using discrete-event simulation. The present work discusses (i) challenges towards building testbeds that allow researchers and practitioners to validate and evaluate the performance of energy-aware traffic-engineering strategies and (ii) requirements when porting simulations to testbeds. We discuss a proof-of-concept platform and an application that use and provide Software-Defined Network (SDN) services created on the Open Network Operating System (ONOS) to validate previously proposed energy-aware traffic engineering strategies. We detail the platform and illustrate how it has been used for performance evaluation.

1 Introduction

Advances in network and computing technologies have enabled a multitude of services — *e.g.* those used for big-data analysis, stream processing, video streaming, and Internet of Things (IoT) [1] — that are hosted at one or multiple data centres often interconnected by high-speed optical networks. Many of these services follow business models such as cloud computing [2], which allows a customer to rent resources from a cloud and pay only for what is consumed. Although these models are flexible and benefit from economies of scale, the increasing amount of data transferred over the network requires continuous expansion of installed capacity in order to handle peak demands. Existing work argues that the amount of electricity consumed by network infrastructure can become a bottleneck and further limit the Internet growth [3].

Given that high performance wired networks are seldom fully utilised, many organisations attempt to curb their energy consumption by reducing the number of resources that are made available during off-peak periods. Several technologies have been employed generally resulting in overall lower energy use; *e.g.* putting resources into low power consumption modes [4], adapting links' data transmission rates [5,6], and grouping and transferring packets in bursts [7]. Traffic engineering [8], initially conceived to enable quality of service and service differentiation, has been investigated as a network-wide approach to improve energy efficiency by, for instance, redirecting traffic and freeing network links that are

S. Guo et al. (Eds.): TridentCom 2016, LNICST 177, pp. 14–23, 2017.
DOI: 10.1007/978-3-319-49580-4_2

henceforth put into low power consumption modes [9,10]. The already difficult traffic-engineering problem of optimising the use of network resources becomes even more challenging when considering energy efficiency.

To simplify configuration and management operations, traffic-engineering schemes are increasingly relying on SDN as it separates control and data planes thus providing a centralised view of (i) the network topology, (ii) running applications and, (iii) traffic demands; which are important requirements to program a network and change its topology according to traffic conditions. In previous work [10,11], we investigated SDN enabled traffic engineering to redirect data flows and reduce energy consumption. The proposed techniques have been evaluated using a discrete-event simulation tool [12] since experimenting with production networks is rarely possible. Although very promising results have been obtained, there is always a need for designing proofs of concept that help evaluating the performance of energy-aware traffic-engineering techniques that support findings of simulations and eliminate undesired biases that may have resulted from simplifying the evaluated scenario.

This work describes challenges and requirements towards building testbeds for evaluating energy-aware traffic engineering strategies and porting simulations to such testbeds as SDN services. We discuss the design and implementation of an SDN application that uses segment-routing and energy-aware algorithms to redirect flows in backbone networks and free certain links [10]. We describe how a custom platform termed as GrEen Traffic engineering testBed (GETB) is used for evaluating the proposed strategies.

The rest of this paper is organised as follows. Section 2 discusses energy-aware traffic engineering, requirements for platforms used for evaluation and SDNs. The testbed used for building proofs of concept is presented in Sect. 3. The SDN application developed for validating and evaluating the performance of the traffic-engineering strategies, its life cycle and results are described in Sect. 4. Section 5 discusses related work and Sect. 6 concludes the paper.

2 Energy-Aware Traffic Engineering and SDNs

Internet traffic engineering deals with issues of performance evaluation, optimisation, and deployment of technology for measuring, characterising, modelling and controlling network traffic. One of its goals is to control and optimise the routing function, to steer traffic through the network in an effective way [8], generally to provide Quality of Service (QoS) and efficient use of network resources. Over the years, interest has grown on applying traffic engineering as a network-wide technique to improve the energy efficiency of network resources [9,13,14]; such efforts are hereafter termed simply as Green Traffic Engineering (GreenTE). Although obtained results are promising, much of the work remains based on numerical analyses and simulation. By attempting to validate our findings using a real testbed, we identified certain GreenTE requirements that experimental platforms should provide, some of which are summarised in Table 1.

The requirements are grouped in hardware resources, information about traffic, energy-optimisation mechanisms, protocols for enabling traffic engineering,

Table 1. GreenTE requirements and commonly adopted approaches.

GreenTE requirements	How requirements are tackled by solutions	
	Simulation	Testbeds
Hardware resources	Simplified and approximate software abstractions of hardware, energy consumption, access time to resources	Often real equipments running in a controlled environment
Traffic information	Commonly assumed that information about flows can be gathered without perturbing the network; centrally available	Monitoring protocols coexist with other network functions, excessive monitoring can impact normal traffic when sharing network resources
Energy-optimisation mechanisms (*e.g.* Link/port switch on/off, Adaptive Link Rate (ALR), Low Power Idle (LPI))	Simplified models, assumptions made when implementing support on simulators, parameter details not always available	Actual ALR and LPI, simulated or actual link/port switch off/on
Network protocols (*e.g.* MPLS-TE, RSVP, SPRING, OpenFlow)	Partial implementation of evaluated schemes, often relying on lower-level protocols that present already approximate behaviour	Normally complete protocol stack, presence of side-effects that may be neglected by simulation tools
Management and control	Commonly assumed that the overhead of configuration and control is negligible	Either dedicated infrastructure allocated to management or it shares resources used by normal traffic; overhead can be measured
Monitoring of power consumption and performance evaluation	Monitoring is performed by gathering stats derived from consumption models	Use of managed PDUs, wattmeters for measuring the consumption of power lines, infrastructure for gathering energy consumption stats

management and control, and measurement of power consumption and performance evaluation. Ideally, modelling and simulation should reflect the behaviour of a real system, but Table 1 provides some assumptions and simplifications found in literature and how they could be circumvented by using an actual testbed. Whilst some elements may look obvious, it is important to notice that testbeds and actual measurements of performance and energy-consumption can eliminate undesirable biases introduced during modelling and can reveal side-effects of solutions not captured during simulations.

Moreover, one of the important requirements of traffic-engineering comprises the ability to gather information about the state of the network, the needs of applications, and configure the behaviour of network resources to steer flows accordingly. Such functions, embedded into data and control planes, were traditionally performed in a decentralised manner, but more recently many traffic-engineering schemes have considered the centralisation of control functions enabled by technologies such as SDNs. SDN separates control and data planes, which in practical terms means that network devices can perform tasks

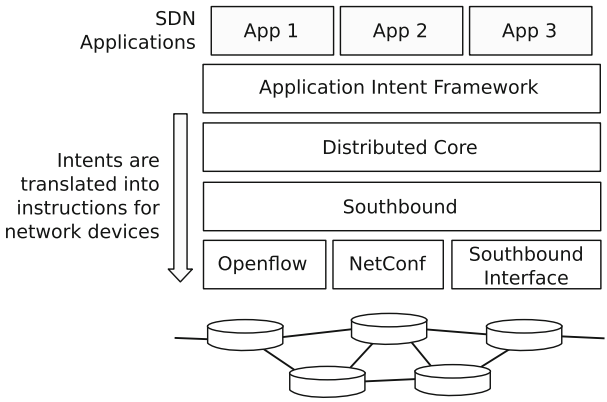

Fig. 1. ONOS intent framework.

that ensure data forwarding (*i.e.* the data plane) whereas management activities (*i.e.* the control plane) are factored out and placed at a central entity termed as SDN controller. SDN has evolved from several technologies, such as OpenFlow, which aim to provide a remote controller with the power to modify the behaviour of network devices via well-defined *forwarding instructions*. Effort has been made towards standardising the interface between controller and the data plane, generally termed as *southbound* API, and the manner the controller exposes network programmability features to applications, commonly called *northbound* API.

SDNs simplify many of the traffic-engineering requirements on gathering traffic information, performing management and control. As described in the next section, we use ONOS, an initiative to build an SDN controller that relies on open-source software components, provides northbound abstractions, and has southbound interfaces to handle OpenFlow capable and legacy devices [15]. In addition to a distributed core that enables control functions to be executed by a cluster of servers, ONOS provides two interesting northbound abstractions, namely the *Intent Framework* and the *Global Network View*. The intent framework, depicted in Fig. 1, allows an application to request a network service without knowledge of how the service is performed. An intent manifested by an application is converted into a series of rules and actions that are applied to network equipments. An example of intent is setting up an optical path between switches A and B with amount C of bandwidth. The global network view, as the name implies, provides an application with a view of the network and APIs to program it. The application can treat the view as a graph and perform several tasks such as finding shortest paths that are crucial to traffic engineering. ONOS provides an application that partially implements SPRING, a framework to enable segment routing currently being standardised by IETF[1]. SPRING pro-

[1] Source Packet Routing in Networking – Working Group https://tools.ietf.org/wg/spring/.

vides features for traffic engineering as it enables an application to specify paths for data flows while avoiding certain network links.

3 GrEen Traffic Engineering TestBed (GETB)

This section describes GETB and how it is used to evaluate energy-aware traffic engineering strategies. Figure 2 illustrates the platform and its main components, depicting the deployment of a set of switches, an SDN controller and applications. The platform comprises components that are common to other infrastructure set up for networking research [16–18]. Moreover, we attempt to employ software used at the Grid5000 testbed [19][2] to which we intend to integrate the platform.

To use the platform, a user requests: a slice or set of cluster nodes to be used by an application, as virtual switches, or serving as traffic sources and sinks; the OS image to be deployed; and the network topology to be used (step 1). We crafted several OS images so that nodes can be configured as SDN controllers and OpenFlow software switches, as discussed later. A bare-metal deployment system copies the OS images to the nodes and configures them accordingly [20], whereas a Python application sets up VLANs and ports of the optical switches in order to form the user-specified network topology.

Once the nodes and network topology are configured, a user deploys her application (step 2 in Fig. 2). All cluster nodes are connected to enclosure Power Distribution Units (ePDUs)[3] that monitor the power consumption of individual sockets [21]. The information on power consumption can be used to evaluate the efficiency of an SDN technique (step 3). The data plane comprises two types of OpenFlow switches, namely software-based and hardware-assisted. The former

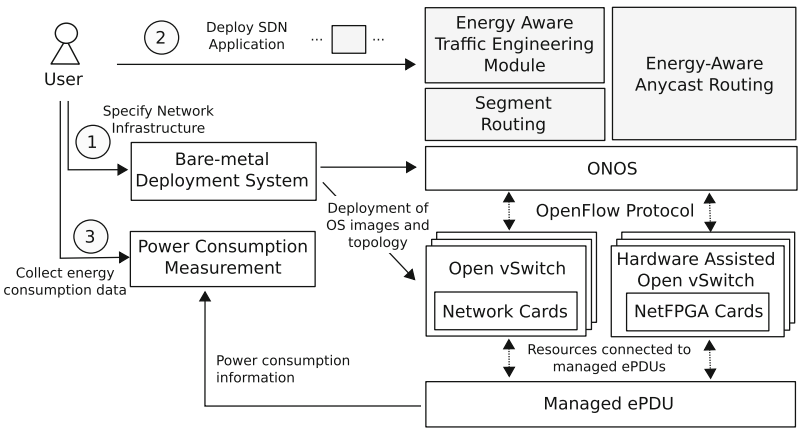

Fig. 2. Overview of the GETB platform.

[2] https://www.grid5000.fr.
[3] http://www.eaton.com/Eaton/index.htm.

consists of vanilla Open vSwitch (OVS) [22], whereas the latter OVS offloads certain OpenFlow functionalities to NetFPGA cards [23] [4]. We use a custom OpenFlow implementation for NetFPGAs initially provided by the Universität Paderborn (UPB) [24] that performs certain OpenFlow functions in the card; *e.g.* flow tables, packet matching against tables, and forwarding.

A NetFPGA card, programmed by default to assist the custom OVS, can allow for other implementations. The current platform comprises ten servers, of which five are equipped with NetFPGA cards and the rest have 10Gbps Ethernet cards with 2 SPF+ ports each and multiple 1Gbps Ethernet ports. The servers are interconnected by both a Dell N4032F optical switch and a Dell N2024 Ethernet switch, which enable testing multiple network topologies.

The infrastructure and the use of ONOS satisfy some requirements of energy-aware traffic engineering namely providing actual hardware, allowing for traffic information to be gathered, using actual network protocols, enabling the overhead of control and management to be measured, and monitoring the power consumption of equipments. Some energy-optimisation mechanisms, however, are still emulated, such as switching off/on individual switch ports. Although the IP cores of the Ethernet hardware used in the NetFPGA cards enable changing the state of certain components, such as switching off transceivers, that would require a complete redesign of the employed OpenFlow implementation. It has been therefore left for future work.

4 Segment-Routing Service

Our strategies for routing data flows so that underutilised links can be freed and powered off [10] stem from the observation that networks are seldom highly utilised, and that most traffic often follows diurnal and weekly patterns. The SPRING framework is used because unlike MultiProtocol Label Switching (MPLS)-TE, link and switch IDs, called Segment Identifiers (SIDs) under SPRING, are global within an autonomous domain, hence allowing for source-routing. At an ingress router a flow can be classified and steered through a given path. This section describes the service life cycle and discusses issues that the testbed enables us to identify and investigate.

4.1 Service Life Cycle

The service, which is a custom version of ONOS segment-routing application, uses a series of ONOS components, including its topology information, flow-rule services, and traffic flow objectives. As shown in Fig. 3, when first launched, a service *Manager* triggers the creation of remaining components. The energy-aware module, which comprises the proposed traffic-engineering algorithms, registers a flow-rule listener in order to measure flow and link utilisations. The configuration component loads a file that specifies how switches are connected

[4] http://netfpga.org/site/#/systems/3netfpga-10g/details/.

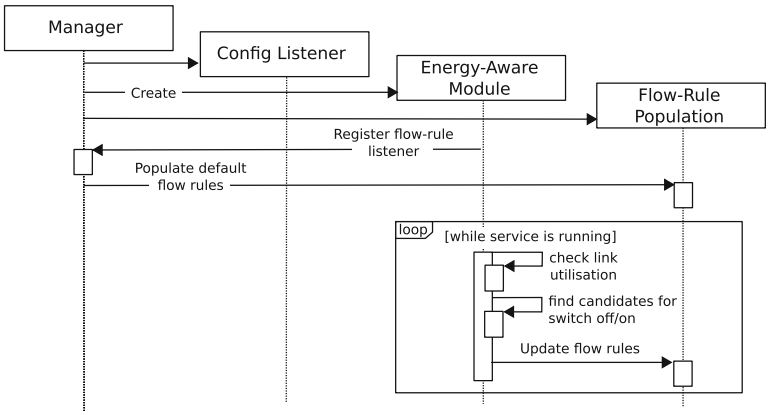

Fig. 3. Start phase of the segment-routing application.

to local networks; information which is then augmented by a topology discovery process. Once the topology is updated, default shortest-path rules are created to guarantee that hosts from a network connected to a switch can reach hosts linked to another switch. A rule consists of a forwarding objective comprising a traffic selector and a treatment. Selectors and treatments result in sets of OpenFlow instructions that are passed to the switches. MPLS push/pop forwarding objectives are created for switches that do not have ports in the source and destination segments — *i.e.* are neither ingress nor egress switches — and normal IP forwarding objectives are built otherwise. While the service is running, the energy-aware module is notified about changes in topology as well as link utilisation, and periodically evaluates whether there are links to switch off/on. If changes in the link availability are required, the energy-aware module requests a flow-rule update to the Flow-Rule Population module.

4.2 GreenTE Issues

Although switching off underused links can be effective from an energy efficiency perspective, sudden bursts in traffic can lead to congestion, hence requiring off links to be made available. In our previous work [11], we proposed algorithms that can react rapidly to traffic bursts by switching links back on when traffic increases. Performance evaluation using discrete-event simulation and UDP-like traffic has shown that the approach can react to traffic bursts without incurring considerable packet loss. It is assumed, however, that the SDN controller can gather the information on link utilisation from switches every second and that a decision on switching a link on can be made and enforced quickly.

We performed a simple test and measured the time taken for a controller to decide on switching on a link. A small network topology was considered as depicted in Fig. 4, which also shows the ONOS graphical interface and a data flow (green lines). The network starts with only a spanning tree turned on and

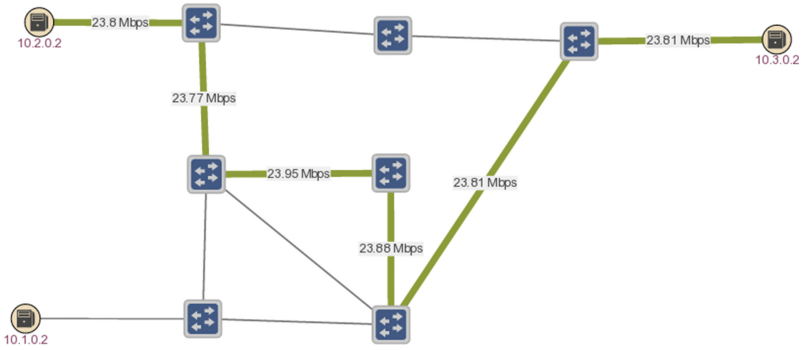

Fig. 4. ONOS GUI showing a data flow avoiding the shortest path.

a TCP flow is injected nearly exceeding the utilisation threshold, above which the controller decides to turn on more links to handle congestion. A second flow is injected, thus exceeding the threshold and forcing the controller to switch links on; we measure the time from flow injection to a switch-on decision. In the simulation, the decision takes on average 1.075 s, with most of the time spent gathering information on link utilisation. In the testbed, the time is on average 20% higher than on simulation.

Other issues that we are investigating concern the stability of the algorithms and the impact of traffic re-routing on TCP flows. Unlike traditional networks where changes in link availability are sporadic, under GreenTE frequent changes can become the rule. Re-routing TCP flows, however, can lead to serious performance degradation due to segments arriving out of order, which can in turn result in multiple duplicate ACKs and trigger the TCP congestion algorithms at the source. We are evaluating how often such conditions can emerge and investigating mechanisms to handle them.

5 Related Work

Several solutions have been proposed to make networks more energy efficient, comprising improvements in used materials, encoding and decoding techniques, power efficient transceivers and other network equipments. Whilst our algorithms can benefit from improvements in hardware and transmission, we focus on techniques that operate at the routing level. In this area, solutions range from putting network interfaces into sleep mode [4] to increasing idle periods of certain links by changing flow paths [9]. A detailed review of the state of the art on this topic is presented in previous work [10].

In the present work, we focused on describing the importance of a platform to evaluate energy-aware traffic-engineering algorithms. Infrastructure for research and development of distributed systems have been established over the years [19,25,26], including platforms for SDN solutions [27] and SDN testbeds [16–18,28]. Our approach has many similarities with previously described platforms,

but we focus on providing an infrastructure that can be used for evaluating both SDN-based solutions and their energy efficiency.

6 Conclusions

This paper discussed an SDN platform for validating and evaluating energy-aware traffic-engineering algorithms. We presented an SDN application that uses segment routing to reroute traffic, and free certain network links that can be switched off. We illustrated the use of the testbed and discussed challenges on improving the stability of routing algorithms and TCP flows on networks employing GreenTE mechanisms.

Acknowledgments. This work is financially supported by the CHIST-ERA STAR project [29].

References

1. Atzori, L., et al.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
2. Armbrust, M., et al.: Above the clouds: a Berkeley view of cloud computing. Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, USA, Technical report UCB/EECS-2009-28, February 2009
3. Kilper, D.: Energy challenges in access, aggregation networks. In: Symposium on Communication Networks Beyond the Capacity Crunch. The Royal Society, London, UK, May 2015. https://royalsociety.org/events/2015/05/communication-networks/
4. Gupta, M., Singh, S.: Greening of the internet. In: ACM Conference on Applications, Technologies, Architectures, Protocols for Computer Communications, ser. SIGCOMM 2003, pp. 19–26. ACM, New York (2003)
5. Gunaratne, C., et al.: Reducing the energy consumption of ethernet with adaptive link rate (ALR). IEEE Trans. Comput. **57**(4), 448–461 (2008)
6. Miyazaki, T., et al.: High speed 100GE adaptive link rate switching for energy consumption reduction. In: International Conference on Optical Network Design and Modeling (ONDM 2015), pp. 227–232, May 2015
7. Nedevschi, S., et al.: Reducing network energy consumption via sleeping, rate-adaptation. In: 5th USENIX Symposium on Networked Systems Design, Implementation, ser. NSDI 2008, pp. 323–336. USENIX Association, Berkeley (2008)
8. Awduche, D., et al.: Overview, principles of internet traffic engineering. RFC 3272 (Informational), Internet Engineering Task Force, May 2002. http://www.ietf.org/rfc/rfc3272.txt
9. Vasić, N., Kostić, D.: Energy-aware traffic engineering. In: 1st International Conference on Energy-Efficient Computing, Networking, ser. e-Energy 2010, pp. 169–178. ACM, New York (2010)
10. Carpa, R., et al.: Segment routing based traffic engineering for energy efficient backbone networks. In: IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS 2014), pp. 1–6, December 2014

11. Carpa, R., de Assuncao, M.D., Glück, O., Lefevre, L., Mignot, J.-C.: Responsive algorithms for handling load surges and switching links on in green networks. In: IEEE International Conference on Communications (ICC 2016), Kuala Lumpur, Malaysia, May 2016
12. OMNeT++ Discrete Event Simulator. https://omnetpp.org/
13. Zhang, M., et al.: GreenTE: power-aware traffic engineering. In: 18th IEEE International Conference on Network Protocols (ICNP 2010), pp. 21–30, October 2010
14. Borylo, P., et al.: Anycast routing for carbon footprint reduction in WDM hybrid power networks with data centers. In: IEEE International Conference on Communications (ICC 2014), pp. 3714–3720. IEEE (2014)
15. Introducing ONOS: a SDN network operating system for service providers. Open Networking Lab ON.Lab, Whitepaper, November 2014. http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf
16. Kim, J., et al.: Proceedings of the Asia-Pacific advanced network. In: OF@TEIN: An OpenFlow-Enabled SDN Testbed over International SmartX Rack Sites, vol. 36, pp. 17–22 (2013)
17. Melazzi, N., et al.: An openflow-based testbed for information centric networking. In: Future Network Mobile Summit (FutureNetw 2012), pp. 1–9, July 2012
18. Sallent, S., Abelém, A., Machado, I., Bergesio, L., Fdida, S., Rezende, J., Azodolmolky, S., Salvador, M., Ciuffo, L., Tassiulas, L.: FIBRE project: Brazil and Europe unite forces and testbeds for the internet of the future. In: Korakis, T., Zink, M., Ott, M. (eds.) TridentCom 2012. LNICSSITE, vol. 44, pp. 372–372. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35576-9_33
19. Bolze, R., et al.: Grid'5000: a large scale and highly reconfigurable experimental Grid testbed. Int. J. High Perform. Comput. Appl. **20**(4), 481–494 (2006)
20. Jeanvoine, E., et al.: Kadeploy3: efficient and scalable operating system provisioning. USENIX Login **38**(1), 38–44 (2013)
21. Rossigneux, F., et al.: A generic and extensible framework for monitoring energy consumption of OpenStack clouds. In: SustainCom, pp. 696–702, December 2014
22. Pfaff, B., et al.: The design and implementation of open vSwitch. In: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2015) (2015)
23. Netfpga 10g. http://netfpga.org
24. The netfpga-10g upb openflow switch. https://github.com/pc2/NetFPGA-10G-UPB-OpenFlow
25. Peterson, L., et al.: PlanetLab architecture: an overview. In: PlanetLab Consortium, Princeton, USA, Technical report PDN-06-031, May 2006
26. GENI: Exploring networks of the future. http://www.geni.net
27. Banikazemi, M., et al.: Meridian: an SDN platform for cloud network services. IEEE Commun. Mag. **51**(2), 120–127 (2013)
28. Ooteghem, J.V., et al.: Sustaining a federation of future internet experimental facilities. International Telecommunications Society (ITS). Technical report 101436 (2014)
29. CHIST-ERA SwiTching And tRansmission (STAR) Project. http://www.chistera.eu/projects/star