Designing and Building SDN Testbeds for Energy-Aware Traffic Engineering Services

Marcos Dias de Assuncao, Radu Carpa, Laurent Lefèvre, Olivier Glück, Piotr Borylo, Artur Lason, Andrzej Szymanski, Michal Rzepka

Noname manuscript No.
(will be inserted by the editor)

Designing and Building SDN Testbeds for Energy-Aware Traffic Engineering Services

Marcos Dias de Assunção[1] · **Radu Carpa**[1] · **Laurent Lefèvre**[1] · **Olivier Glück**[1] · **Piotr Boryło**[2] · **Artur Lasoń**[2] · **Andrzej Szymański**[2] · **Michał Rzepka**[2]

the date of receipt and acceptance should be inserted later

Abstract As experimenting with energy-aware techniques on large-scale production infrastructure is prohibitive, a large number of proposed traffic-engineering strategies have been evaluated only using discrete-event simulations. The present work discusses (i) challenges towards building testbeds that allow researchers and practitioners to validate and evaluate the performance and quality of energy-aware traffic-engineering strategies, (ii) requirements to fulfill when porting simulations to testbeds, and (iii) two proof-of-concept testbeds. One testbed uses and provides Software-Defined Network (SDN) services created on the Open Network Operating System (ONOS) while the other is a composition of virtual Open vSwitches (OVS) controlled by the Ryu SDN framework. The aim of the testbeds is to validate previously proposed energy-aware traffic engineering strategies in different environments. We detail the platforms and illustrate how they have been used for performance evaluation. Additionally, the paper compares results obtained in the testbeds with evaluations performed using discrete-event simulations and presents challenges faced while implementing energy-aware traffic engineering mechanisms as SDN services in testbed environments.

Keywords energy-awareness · segment routing · anycast routing · testbeds

[1]Inria Avalon, LIP Laboratory, École Normale Supérieure de Lyon, University of Lyon, France
[2] AGH University of Science and Technology, Department of Telecommunications, Krakow, Poland

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) [10] — that are hosted at one or multiple data centres often interconnected with high-speed optical networks. Many of these services follow business models such as cloud computing [8], which allow a customer to rent resources from a cloud and pay only for what is really 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, however, that the amount of electricity consumed by network infrastructure may become a bottleneck and further limit the Internet growth [26].

Given that high performance wired networks are seldom fully utilised, many organisations attempt to curb their energy consumption by reducing the amount of resources that are active during off-peak periods. Several technologies have been employed for this purpose, *e.g.* putting resources into low power consumption modes [23], adapting links' data transmission rates [22,30], and grouping and transferring packets in bursts [31]. Their utilisation, in general, results in lower overall energy use. On the other hand, traffic engineering [11], initially created to reduce network bottlenecks by shifting traffic to underutilised links, has been also investigated as a network-wide approach to improve energy efficiency by, for instance, consolidating traffic on a limited number of links, and thus enabling the remaining links to enter the low power consumption modes [41,19]. As a result, the already difficult problem of optimising the use of network resources became even more challenging.

To simplify configuration and management operations, traffic-engineering schemes are increasingly relying on SDN as it separates control and data planes and provides 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 virtual topology according to traffic conditions. In our previous work [19,18,9], we investigated SDN enabled traffic engineering to redirect data flows and reduce energy consumption. We also considered SDN based energy-aware anycast routing [14] extended by introducing cloud services differentiation [15], models of cooperation between SDN controller and cloud orchestration software [17] and the concept of interplay between fog and cloud infrastructures supported by wide area software defined networking [16]. As experimenting with production networks is rarely possible, the proposed techniques have been evaluated using a discrete-event simulation tool (OMNet++ [5]), which provided very promising and valuable results. However, to fully investigate the subject and assess whether simplifications made during simulations have not led to biased results, it is necessary to design proof of concept implementations by using testbeds.

This work describes challenges and requirements towards building platforms for evaluating energy-aware traffic engineering applications and porting simulations to such testbeds as SDN services. We discuss the design and implementation of two approaches: an SDN application that uses segment-routing to redirect flows in backbone networks in order to free certain links [19] and energy-aware anycast strategies considering the type of energy used to power target data centres. We describe how custom platforms, called The GrEen Traffic engineering testBed - Segment Routing (GETB-SR) and The GrEen Traffic engineering testBed - Anycast Routing (GETB-AR), are used for evaluating the proposed applications. We compare results gathered in the testbeds with simulation-based results and present challenges faced while designing energy-aware traffic engineering mechanisms as SDN services in testbed environments.

The rest of this paper is organised as follows. Section 2 discusses energy-aware traffic engineering requirements for platforms used for evaluation and SDN concepts. The testbeds used for building proofs of concept are presented in Section 3. The SDN applications developed for validating and evaluating the performance of the traffic-engineering strategies, their life cycles, results and issues regarding deployment in the testbed are described in Section 4. Section 5 discusses related work and Section 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 [11], providing appropriate 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 [41,43,14]; 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. In an attempt 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 2.

The requirements are grouped in hardware resources, information about traffic, energy-optimisation mechanisms, protocols for enabling traffic engineering, 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 2 provides some assumptions and simplifications found in literature. Whilst some elements may look obvious, many of them may affect the overall assessment of the analysed solution. Testbeds and actual measurements of performance and energy-consumption may eliminate most of them and may reveal side-effects not captured during simulations.

An important requirement of traffic-engineering comprises the ability to gather information about the state of the network, the needs of applications, source of electricity used to power network elements (*i.e.* renewable, non-renewable) and to configure the behaviour of network elements to steer traffic 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 perform tasks 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 the 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

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

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. An example of such an application is cloud infrastructure utilising a network to provision cloud services for customers. In this case the *northbound* interface may carry information about availability of computing resources or renewable energy in multiple data centres.

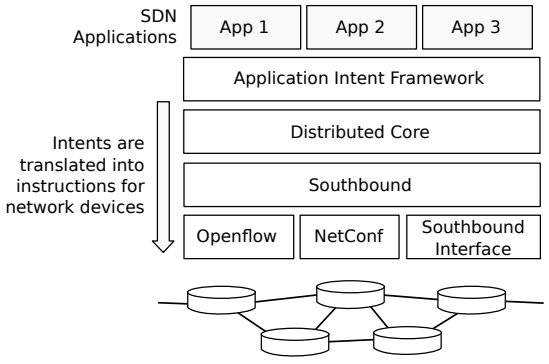

Fig. 1 ONOS Intent Framework.

SDNs simplify many of the traffic-engineering requirements on gathering traffic information, performing management and control. As described in the next section, in the GETB-SR 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 [7]. 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 Figure 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 devices. 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 may treat the view as a graph and perform several tasks that are crucial to traffic engineering, such as finding shortest paths. ONOS provides an application that partially implements SPRING, a framework to enable segment routing currently being standardised by IETF[1]. SPRING provides features for traffic engi-

[1] https://tools.ietf.org/wg/spring/

neering as it enables an application to specify paths for data flows while avoiding certain network links.

Simultaneously, in the GETB-AR we use Ryu, a component-based software defined networking framework aimed at creating new network management and control applications [6]. It provides *southbound* API to control network equipment using various protocols (OpenFlow versions 1.0, 1.2 1.3, 1.4 and 1.5, Netconf or OF-config). Ryu also defines *northbound* API for deploying SDN applications such as energy-aware traffic engineering mechanisms. Applications in Ryu are software entities running in individual threads and sending asynchronous events to one another [36]. The Ryu controller was chosen for the GETB-AR testbed as an alternative solution to the ONOS framework used for GETB-SR and to acquire wide experience on configuration and implementation of various SDN controllers. Each application in Ryu has a dedicated FIFO queue to hold incoming events while appropriate event handlers are called for various event types. The application programming model is depicted in Figure 2. The potential of the Ryu framework is proved by commercial deployments in NTT proprietary cloud data centres.

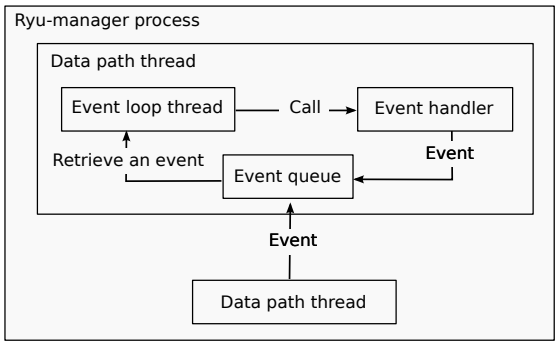

Fig. 2 Application architecture within Ryu framework [36].

3 Proof-of-Concept Platforms

We built two proof-of-concept testbeds: GrEen Traffic engineering testBed - Segment Routing (GETB-SR) and The GrEen Traffic engineering testBed - Anycast Routing (GETB-AR). The aim of both testbeds is to thoroughly investigate energy-aware traffic engineering approaches proposed in our previous work but evaluated only using discrete-event simulation. Additionally, based on the conducted research, we present challenges faced while implementing energy-aware traffic engineering mechanisms as SDN services in testbed environment.

3.1 GrEen Traffic engineering testBed – Segment Routing (GETB-SR)

Figure 3 illustrates the GETB-SR platform and its main components, depicting the deployment of switches, an SDN controller and applications. At smaller scale, the platform comprises components that are common to other infrastructures set up for networking research [27, 29, 37]. Moreover, we attempt to employ software used at the Grid5000 testbed [13][2] to which we intend to integrate the platform.

To use the platform, a user requests: a slice or a set of cluster nodes to be used by an application as virtual switches or serving as traffic sources and sinks, an OS image to be deployed and a 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 is used to copy the OS images to the respective nodes and configure them accordingly [25], whereas a Python application configures VLANs and interfaces of the virtual switches emulating optical switches interconnecting the nodes in order to create the user-specified network topology.

Once the nodes and the network topology are configured, the user deploys his or her application (step 2 in Figure 3). All cluster nodes are connected to enclosure Power Distribution Units (ePDUs)[3] that monitor the power consumption of individual sockets [35]. This information on power consumption may 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 consists of a vanilla OVS [34], whereas the latter OVS offloads certain OpenFlow functionalities to NetFPGA cards [3][4]. We use a custom OpenFlow implementation for NetFPGAs, initially provided by the Universität Paderborn (UPB) [4], that performs certain OpenFlow functions in the card, *e.g.* flow tables, packet matching against tables, and forwarding.

Although the NetFPGA cards are by default programmed as custom OpenFlow switches, a user can reprogram them for different purposes by copying a bitstream file to their flash memories and rebooting the system. The current testbed comprises eight servers – five Dell R720 servers equipped with a 10Gbps Ethernet card with 2 SPF+ ports each and three HP Z800 servers with NetFPGA cards with 4 SPF+ ports each. All servers also have multiple 1Gbps Ethernet ports.

[2] https://www.grid5000.fr
[3] http://www.eaton.com/Eaton/index.htm
[4] http://netfpga.org/site/#/systems/3netfpga-10g/details/

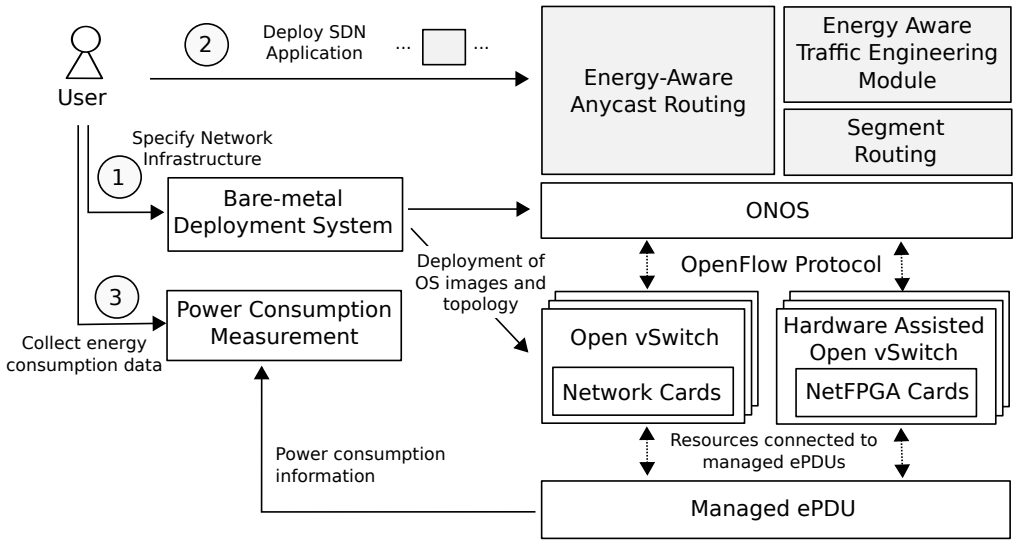

Fig. 3 Overview of the GETB-SR platform.

The SPF+ ports have optical transceivers and are all interconnected by a Dell N4032F L3 switch whereas two 1Gbps Ethernet ports of each server are connected to a Dell N2024 Ethernet switch. The platform is depicted in Figure 4. This configuration enables testing multiple network topologies.

Fig. 4 The GETB-SR platform.

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 equipment. 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 therefore been left for future work.

3.2 GrEen Traffic engineering testBed - Anycast Routing (GETB-AR)

The GETB-AR testbed was placed on an 8-core 2.83GHz server running VMware ESXi 5.5, with maximum shared allocation of 20GB RAM for all machines. Virtual disks of the testbed machines were placed on a local RAID 10 array of 4x15k RPM disks. All machines run with 512MB vRAM, except for the VM hosting the SDN controller, which has 2048MB vRAM. VM disks are thin-provisioned 16GB linked clones. There are 14 machines running OVS switches (to support NSFNet topology and validate results presented in previous work [14]). All machines are connected with one vNIC to a management network allowing for remote access and 7 NICs for testbed connectivity. A set of 21 vSwitches was created for connectivity among OVS machines. An additional set of 14 vSwitches and a set of 14 client machines were created to provide user traffic. A separate set of

14 vSwitches and a set of 14 machines were created to provide simulated Data Centre (DC) facilities. Thus, all the testbed traffic is being exchanged within the single physical server.

A Ryu SDN controller application was deployed to handle traffic in accordance with the energy-aware anycast strategies introduced in previous work [14]. Accurate routing of both anycast and unicast traffic relies on up-to-date network data including topology, available resources and other control information gathered from network nodes, and user input via a REST API interface. The acquired data is used to create a network graph continuously updated by the controller's topology discovery module. When a new connection is initialised, its first packets are forwarded to the controller and thoroughly examined to determine unique flow identifiers based on L3/L4 headers. The best path is calculated using one of the available algorithms taking into account available destination nodes, resource constraints and active routing policy. Flow entries are then inserted into each intermediate node's flow table, causing all subsequent packets that belong to the same flow to be forwarded automatically to an appropriate destination, without further notice of the network controller. The southbound interface uses the OpenFlow protocol version 1.3, enabling cooperation with compliant hardware switches. Figure 5 illustrates the GETB-AR platform. To model optical network properties we used MPLS tunnels that reserve assumed amount of resources on the links along the whole path. The aforementioned mechanisms regard to the management and control as well as traffic information GreenTE requirements listed in Table 2.

Future extensions. In the current GETB-AR testbed configuration, information about the type of power used at each DC is provided manually through the SDN controller northbound API extended in our testbed. Further development is envisioned to acquire information about the percentage of green energy used in all the nodes automatically. This task is similar to the discovery of network topology where Link Layer Discovery Protocol is utilised along with an appropriate SDN controller extension. We consider two approaches to implement this function.

The first one, the Power Grid Orchestration (PGO), assumes that all nodes have an interface to the power grid network, which allows to estimate the share of renewable energy in the energy supplied to the node. This additional information has to be forwarded to the SDN controller with the use of an appropriate Open Flow protocol extension. The SDN controller is supposed to update its database and modify the behaviour of the traffic engineering applications. The main advantage of PGO is the automatic acquisition of information on the power status of all network nodes and possible use of this knowledge for traffic engineering purposes.

The second concept, called the Open Stack Orchestration (OSO), involves SDN integration in the area of green networking with *OpenStack* software. In this case, the information about the power status of all DC nodes will come from the *OpenStack* module responsible for efficient utilisation of the controlled resources. This is possible due to the use of energy-aware interface between SDN and *OpenStack*. The *OpenStack* software may estimate and distribute a synthetic index based on many factors, including the share of green energy consumed in the node, current price of energy, utilisation of computing resources, the number and complexity of computing tasks scheduled in a given time scale and many others. The main disadvantage of this solution is that distribution of a single index value for each DC node might be insufficient for other traffic engineering applications. Both approaches to gather node power conditions are of course not exclusive and may be used together.

Another possible extension of GETB-AR testbed is to implement sample network services. We work towards implementing security mechanisms in a cloud computing infrastructure composed of public and private resources. Our research in the area of secure hybrid cloud infrastructure [28] is coordinated with the GETB-AR design, implementation and configuration. Public cloud infrastructure is composed of all DC nodes established in the GETB-AR testbed, private cloud computing resources are represented by all virtual machines attached to network nodes. Some experiments and evaluation of an ongoing work within the secure green networking testbed are planned for near future.

4 Green Traffic Engineering Use Cases

In this section, we discuss two energy-aware traffic-engineering approaches. The first uses segment routing to reroute traffic in order to free links that henceforth become candidates to be switched off, whereas the second establishes paths for anycast requests considering the type of energy used to power target data centres. A crucial contribution of this section is the comparison of results obtained in the testbeds with those using discrete-event simulations.

4.1 Segment-Routing Service

Our strategies for routing data flows so that underutilised links may be freed and powered off [19] stem

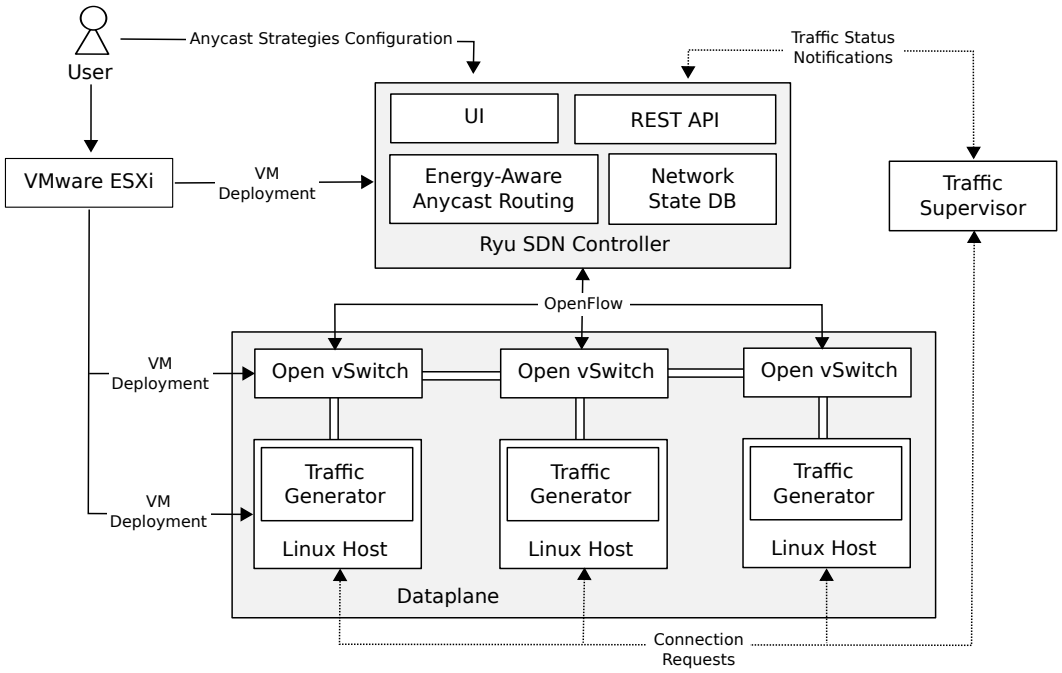

Fig. 5 Overview of the GETB-AR platform.

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 in MultiProtocol Label Switching (MPLS)-TE, link and switch IDs called Segment Identifiers (SIDs), are global within an autonomous domain, hence allowing for source-routing. In this way, a flow may be classified at an ingress router 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.

4.1.1 Service Life Cycle

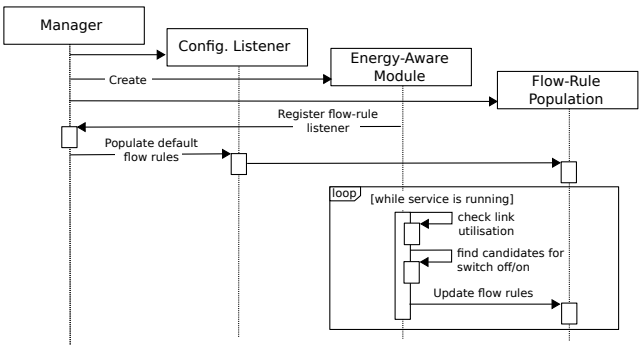

Fig. 6 Start phase of the segment-routing application.

The service, which is a custom version of the ONOS segment-routing application, uses a series of ONOS components, including its topology information, flow-rule services, and traffic flow objectives. As shown in Figure 6, a service *Manager* triggers the creation of remaining components when it is launched. The energy-aware module, which comprises the proposed traffic-engineering algorithms, registers a flow-rule listener to measure flow traffic and link utilisation. The configuration component loads a file that specifies how switches are connected 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 may 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.

4.1.2 GreenTE Issues

Although switching off underused links may be effective from energy efficiency perspective, sudden bursts in traffic may lead to congestion, hence requiring links to be made available. Our previous work [18] proposed algorithms that may 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 successfully reacts to traffic bursts without incurring considerable packet loss. It is assumed, however, that the SDN controller gathers the information about link utilisation from switches every second and that a decision to power a given link back on may be taken and enforced quickly.

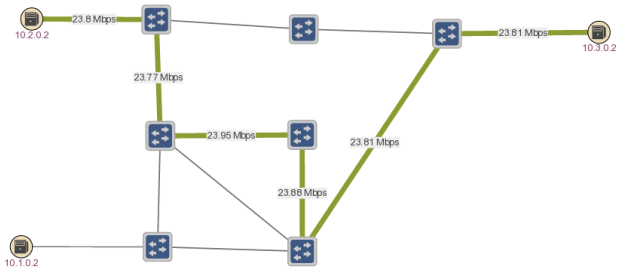

Fig. 7 ONOS GUI showing a data flow avoiding the shortest path.

We performed a simple test and measured the time needed for a controller to decide whether a link should be switched on. A small network topology was considered, as depicted in Figure 7. The Figure also shows the ONOS graphical interface and a data flow (green lines). The network starts with a minimal number of links turned on, forming a spanning tree, and with a TCP flow that nearly exceeds the utilisation threshold, above which the controller decides to turn on more links to handle congestion. A second flow is then 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 seconds, with most of the time spent gathering information on link utilisation. In the testbed, the time is on average 20% higher than in simulation.

We notice that the difference in results between simulation and real testbeds are generally due to simulations assuming zero delay at multiple parts of the processing pipeline and the manner network events are handled. While a single delay simplification would have marginal impact on the results, multiple delays along the packet processing pipeline can account for up to 30% difference in the time to react to changes. Examples of delay simplifications during discrete-event simulations include: instantaneous insertion of forwarding rules into the data path, immediate update of routing tables, fast propagation of flow counters from the simulated hardware ports to the software of the SDN switch, and instantaneous processing of IP UDP/TCP packets. Generally, the only delay properly handled by a simulator is packet queueing time.

Moreover, existing work has already shown that updating the data path forwarding rules is slow in current commercial SDN switches [24]. Google employees report[5] that their SDN-based WAN had an outage due to this issue on propagation of forwarding rules. Improvements can be made in the simulation software to account for some delay, and in the hardware design itself to reduce the time to propagate rules.

Other issues that we investigated 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 may be the rule. Re-routing TCP flows, however, may lead to serious performance degradation due to segments arriving out of order, which in turn result in multiple duplicate ACKs and hence triggering the TCP congestion algorithms at source. Even though the algorithms in the simulator mimic the behaviour of their corresponding theoretical models, they differ from the actual network software implementations provided by certain operating systems. The Linux kernel, for instance, includes several non-standard optimisations [38]. While simulations highlighted that re-routing TCP flows severely impacts the throughput of the transported TCP flows, empirical evaluation on the testbed demonstrated almost no impact under the same conditions. We believe that existing work that wraps real network software stack into simulators[6] may help minimise this issue.

4.2 Energy-Aware Anycast Routing

One of the approaches to green networking is based on using energy produced from renewable energy sources as this results in carbon footprint reduction. This approach is especially important in the context of networks connecting Data Centres (DCs). Providing numerous cloud computing services requires huge computing power, which in turn, results in enormous power requirements by DCs. Therefore, powering DCs with

[5] https://atscaleconference.com/videos/lessons-learned-from-b4-googles-sdn-wan/

[6] http://www.wand.net.nz/~stj2/nsc

renewable energy is expected to reduce CO_2 emission significantly. In hybrid power networks, selected DCs are powered using energy from renewable sources while the rest uses energy obtained from conventional sources. As one of the paradigms of cloud computing is to offer the same services in different DCs at the same time, it is possible to choose one of many possible locations to provide the requested service. A corresponding routing scheme is called anycast [20]. Another important feature of cloud computing is to offer services in an on-demand fashion. That is why incoming requests are unpredictable and the underlying network must be investigated under a dynamic traffic scenario [20]. Therefore, features like global view of the current network topology and allocated paths as well as possibility to dynamically handle incoming requests are essential. Both features are ensured by the SDN idea with a centralised controller. The GETB-AR testbed is used to evaluate anycast strategies that aim to reduce greenhouse gases emission by processing resource-intensive requests in data centres powered from renewable energy sources.

The anycast routing problem consists of a graph $G(V, E)$ representing the physical network, where V is the set of nodes and E is the set of network links. A subset of $V_{DC} \subset V$ comprises data centres that serve user requests and $V_C \subset V$ denotes source switches/routers to which users are connected. V_{gDC} comprises *green* data centres that are powered by renewable energy sources, whereas V_{bDC} consists of *brown* data centres that use traditional energy sources. The following relations are met: $V_{gDC} \bigcap V_{bDC} = \emptyset$, $V_{gDC} \bigcup V_{bDC} = V_{DC}$, $V_{DC} \bigcap V_C = \emptyset$ and $V_{DC} \bigcup V_C = V$. The strategies aim to prioritise the use of V_{gDC} while respecting several connectivity and availability constraints. Figure 8 illustrates the virtual infrastructure and topology developed on the GETB-AR platform.

4.2.1 Anycast strategies

In previous work [14], we proposed three anycast routing strategies (*randomGreen*, *closestGreen* and *closestGreenWithPenalty*) focused on reducing the carbon footprint. All the strategies require very limited additional control information and hence may be easily implemented using a centralised network controller consistent with the idea of SDN. The proposed strategies are compared to the *single* anycast strategy, which is the base reference strategy, well known and widely described in literature. However, the general idea behind the proposed heuristics is to provide some kind of *green* DC preference while choosing a destination for an anycast request and, at the same time, consider multiple DCs as targets for the request to improve network performance. Thus, to ensure a comprehensive assessment of the three proposed strategies, we also provided two additional reference strategies, *random* and *closest*, which consider multiple DCs as targets for an anycast request but do not prefer *green* DCs over *brown* ones. In this paper we provide only short descriptions of those strategies, but detailed information along with pseudocode can be found in previous work [14].

In the *single* strategy an anycast request is served by first randomly choosing a single destination d within V_{DC}. Then it checks whether the wavelength continuity constraint can be met by any of the three alternate optical paths between the source and randomly selected destination node. If a lightpath is available between the source $s \in V_C$ and the destination d, then the request is accepted; otherwise it is rejected. In the *random* strategy the network controller iteratively tries to establish the lighpath between $s \in V_C$ and a random $d \in V_{DC}$. The strategy ends when the first available d is found or none of the possible destinations is available after examining all possible destinations. In case of the *closest* strategy, the closest (in the hop manner) destination d is chosen and the lightpath is established between s and d. If none of $d \in D$ is available then the request is rejected. In the *randomGreen* the SDN controller iteratively tries to establish the lighpath between s and a random $d \in V_{gDC}$. If all $d \in V_{gDC}$ are tried without success then the same scheme is performed for $d \in V_{bDC}$. In the *closestGreen* strategy the network controller performs operations analogous to the *closest* strategy for all $d \in V_{gDC}$. If none of $d \in V_{gDC}$ is available then the SDN controller repeats the same operation for all $d \in V_{bDC}$. If none of $d \in D$ is available then the request is rejected. The *closestGreenWithPenalty* strategy works analogously to the *closest* strategy but with one significant difference. The distance from s to $d \in V_{bDC}$ is multiplied by the *penalty* factor, where the *penalty* is an input parameter of the strategy. The *closestGreenWithPenalty* strategy with $penalty = 1.0$ is equivalent to the *closest* strategy.

To sum up, the three proposed strategies (*randomGreen*, *closestGreen* and *closestGreenWithPenalty*) create an opportunity to reduce carbon footprint by favouring *green* DCs over *brown* ones. However, as a side effect, the average lightpath length and resource utilisation may significantly increase in case of *randomGreen* and *closestGreen* strategies, where green DCs are firmly prioritised. Thus, the *closestGreenWithPenalty* strategy was proposed to flexibly balance the trade-off between carbon footprint reduction and the average lightpath length. The higher value is assigned to the *penalty* parameter the more *green* DCs are preferred over *brown* ones and, at the same time, the higher is the average

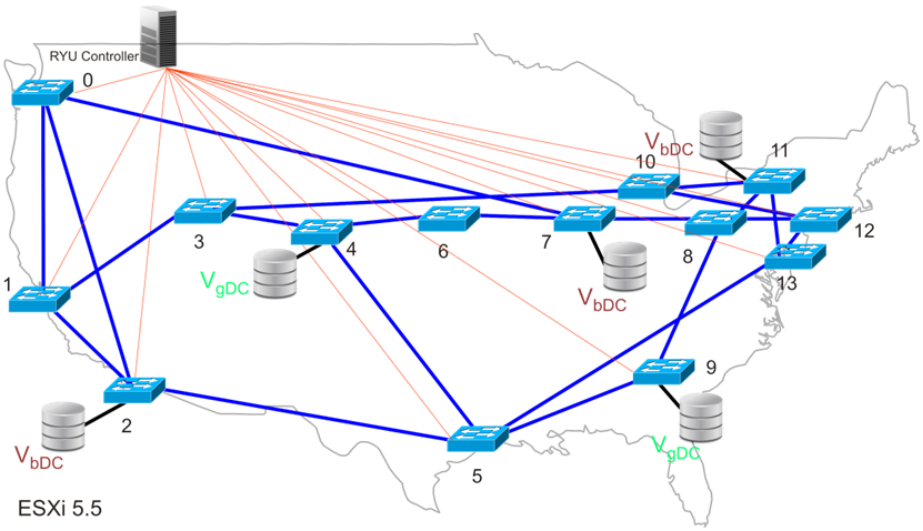

Fig. 8 Overview of the evaluated scenario.

lightpath length. Each of the three proposed strategies is expected to reduce the blocking probability of anycast requests in comparison to the *single* strategy. Thus, for a comprehensive assessment, the impact of the proposed strategies on network performance will be additionally investigated in comparison to the auxiliary reference strategies, *random* and *closest*. The *closest* strategy is especially significant in the context of network performance as it is expected to ensure the shortest average lightpath length and thus the lowest network resource occupancy. We use a simple shortest path (least hop) routing algorithm and firstfit wavelength assignment with k-alternate paths [39].

The implementation of anycast strategies is an essential part of the controller's application and relies on its knowledge of network state and available resources. The network graph created by the topology discovery module comprises vertices and edges representing network nodes and network links. Neighbour adjacency is discovered by exchange of LLDP packets containing unique identifiers. Each vertex holds node-specific data including its type (brown DC, green DC, client) and interface configuration. Information about available and reserved wavelengths is held by graph edges. This information, along with penalty values and flow-specific parameters is utilised in the process of path computation. Once a *packet_in* message is received by the controller, its contents are extracted to determine ingress node and flow parameters, namely IP addresses and TCP/UDP port numbers. After an optimal path, with regard to active anycast strategy, is chosen by the routing algorithm, OpenFlow instructions are passed to each switch along the desired flow route. Each intermediate node is supplied with two flow entries specifying appropriate output ports to enable bidirectional transmission. Once the path is set up, it may be removed anytime, allowing for flexible network reconfiguration.

4.2.2 Results

Two metrics were used to asses the anycast strategies. The first one estimates carbon footprint and is the average ratio of the power consumed from non-renewable sources to the amount of DC traffic switched in all DCs ($brownKiloWatts/(Gb/s)$). Thanks to this normalisation we obtain comparable results under different traffic loads. The second metric is the blocking probability, calculated as the ratio of rejected requests to all requests. An external traffic generator module was implemented to evaluate performance of anycast strategies in the testbed environment. Each of the 14 hosts runs a client application able to generate multiple TCP/UDP streams using the iperf tool. Both anycast and unicast connections are set up on demand while compliance with traffic parameters described in previous work [14] is enforced by a central supervisor generating relevant connection requests. The SDN controller collects statistics of each flow, *e.g.* transmission time, end nodes and assigned resources. Data regarding the complete routing and network state is available to the controller while end of transmission timestamps are obtained directly from end nodes. A dedicated controller module was implemented to gather and store historical data about flows. As a result of detailed flow database analysis, both indicators, *e.g.* $brownKiloWatts/(Gb/s)$ and blocking probability were calculated.

Figures 9 and 10 present carbon footprint of the analysed anycast strategies with different data centres indicated as *green*. In each figure, the left-hand graph

presents results obtained in the discrete-event simulator and the right-hand one shows results obtained in the GETB-AR testbed. Figures 11 and 12 show the corresponding total blocking probabilities.

The main differences between results obtained in discrete-event simulations and GETB-AR testbed are related to the absolute values. Additionally, differences between particular strategies are less pronounced in the testbed comparing to the simulations. The reason behind such observations is a difference in modelling of network resources in both environments. Finally, results obtained in GETB-AR testbed oscillate between different network loads while this effect cannot be observed for simulation-based results. It is a result of more realistic approach to generation of user requests in virtual machines instead of simulating such requests as programming objects in simulation environment. All of the issues indicate that any simulation-based conclusions should always be drawn with proper caution. However, those minor disparities do not affect a general conclusion that preliminary assessment of anycast strategies performed in discrete-event simulations is proved to be true with results obtained in the GETB-AR testbed. Simulation results show that the three proposed strategies *randomGreen*, *closestGreen* and *closestGreenWithPenalty* may significantly reduce CO_2 emission in comparison to all reference strategies, and decrease blocking probability in comparison to the *single* strategy. Furthermore, in each simulation scenario the *closestGreenWithPenalty* strategy retains network performance at a level comparable with the *closest* strategy, which is expected to provide the lowest network utilisation. Additionally, the *penalty* parameter allows the *closestGreenWithPenalty* strategy to elastically balance the trade-off between carbon footprint reduction and network performance.

Several issues not observed in discrete-event simulations needed to be addressed during development and deployment of the testbed. These were mainly related to detection and assessment of network events and limited computation resources available to the controller and switch software causing packet processing delays. While a connection setup is relatively easily detected by the controller when packets with an unknown flow identifier are received, it is difficult to tell whether the connection is still active at a particular moment. Considering a variety of services being present on the network, there are no common measures such as timeouts indicating expired connections. Although TCP flows may be monitored for presence of *FIN* and *RST* flags, such method is not suitable for UDP streams. Still, instant deallocation of unused resources and network state database update is essential for energy-aware routing algorithms to work efficiently. One of the proposed solutions involved applying idle timeouts to handle expired flows. However, such approach could cause several connections to be disrupted prematurely or allocated resources to be freed belatedly. To alleviate the issue, additional controller REST API calls were implemented for client application to notify controller of connection teardown immediately after the end of transmission. This, however, requires client applications to be aware of flow setup mechanism and additional control modules to be implemented by their developers.

Another issue faced in the testbed environment was caused by limited performance of the SDN controller. Although satisfactory switching rates may be achieved by Open vSwitch software deployed on VMWare virtual machines, bursty traffic may drain the controller's resources and make it a network bottleneck. Considering the time required to parse incoming *packet_in* OpenFlow message, calculate efficient path and push flow information to all intermediate flow tables, additional precautions should be taken to prevent inappropriate handling of network streams. This includes additional mechanisms ensuring all flows to be detected only on reception of their initial packet and *packet_out* messages being sent in the correct order. Again, the issue is consistent with GreenTE requirement on management and control presented in Table 2.

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 improvements in network equipment. Whilst our algorithms may 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 [23] to increasing idle periods of certain links by changing flow paths [41]. A detailed review of the state of the art on this topic is presented in previous work [19].

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 [13,33,2], including platforms for SDN solutions [12] and SDN testbeds [27,29,37,32]. Our approach and previously described platforms have many similarities, but we focus on providing an infrastructure that may be used for both evaluating SDN-based solutions and assessing their energy efficiency.

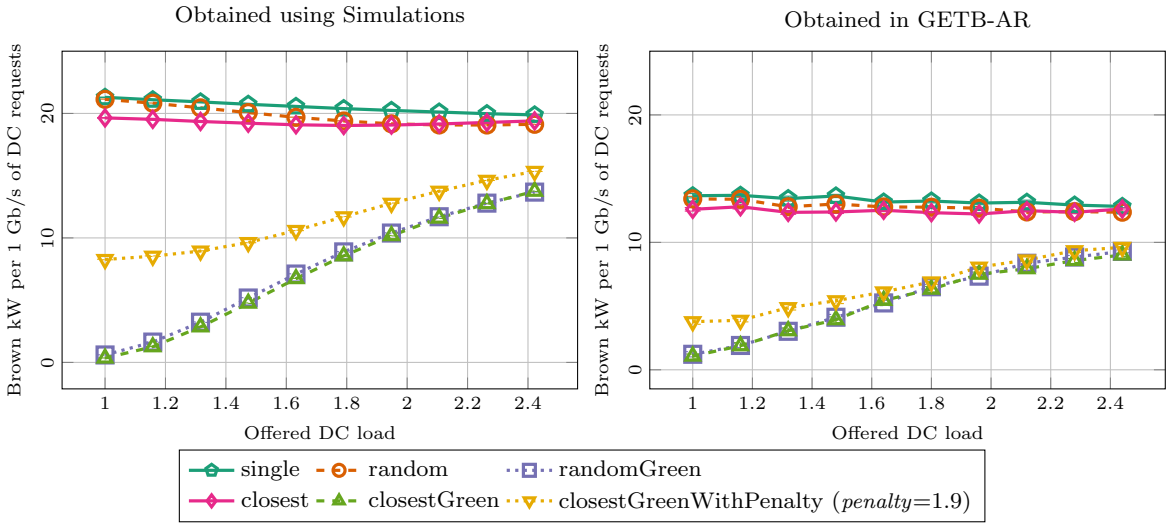

Fig. 9 Brown kilowatts needed to handle 1 Gb/s of DC requests in the NSF network with $V_{gDC} \in \{4, 11\}$.

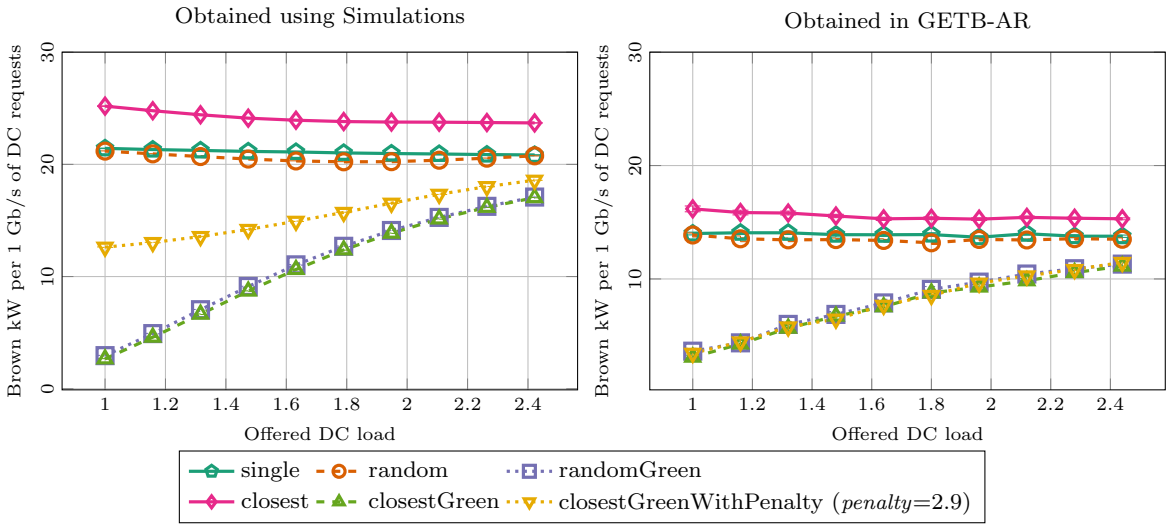

Fig. 10 Brown kilowatts needed to handle 1 Gb/s of DC requests in the NSF network with $V_{gDC} \in \{9, 11\}$.

Most of research in the area of testbed-based evaluation of energy-aware mechanisms is focused on data centres. For example, a green DC testbed has been proposed in previous work [21]. It adjusts the workload handling process to the availability of solar and wind energy and optimises air conditioning with regard to the outside air temperature. A platform for energy-aware analysis of an internal DC network was also proposed [40], where the authors combined hardware with emulation techniques and proposed an extension to the OpenFlow protocol in order to measure energy consumption of infrastructure components as well as link occupancy. Another extension to the OpenFlow protocol was also evaluated in a testbed environment [42], where the aim was to reduce energy consumption by turning off switches and their ports in an internal DC network along with changing clock frequency in the equipment.

6 Conclusions

This paper discussed challenges towards building testbeds and requirements imposed on those testbeds and SDN platforms for validating and evaluating energy-aware traffic-engineering algorithms. We presented two SDN applications: the first one uses segment routing to reroute traffic in order to free certain network links which can then be switched off, whereas the second establishes paths for anycast requests considering the type of energy used to power target data centres. We also illustrated the use of the testbeds. Specifically, in the

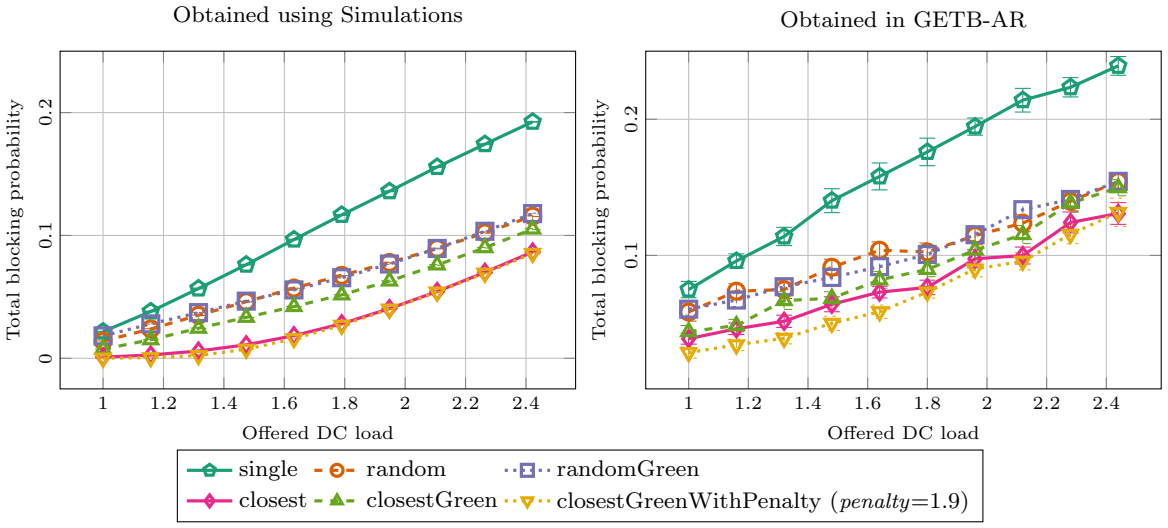

Fig. 11 Total blocking probability of both traffic types in the NSF network with $V_{gDC} \in \{4, 11\}$.

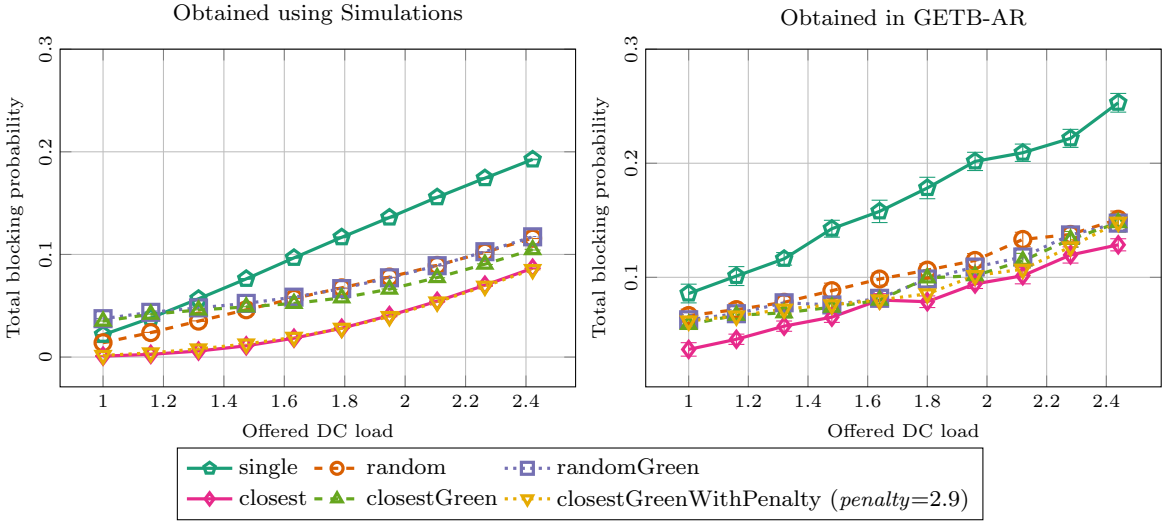

Fig. 12 Total blocking probability of both traffic types in the NSF network with $V_{gDC} \in \{9, 11\}$.

GETB-SR testbed we discussed challenges on improving the stability of routing algorithms and TCP flows in networks employing GreenTE mechanisms, and by utilising the GETB-AR testbed we compared effectiveness of energy-aware anycast strategies evaluated in the testbed and contrasted it with results obtained using discrete-event simulation. We also presented issues specific of testbed environment along with challenges toward deployment of *green* anycast strategies in the GETB-AR.

Acknowledgments

This work was financially supported by the CHIST-ERA ERA-Net SwiTching And tRansmission (STAR) project [1].

References

1. CHIST-ERA STAR (SwiTching And tRansmission) project. http://www.chistera.eu/projects/star
2. GENI: Exploring networks of the future. http://www.geni.net
3. Netfpga 10g. http://netfpga.org
4. The netfpga-10g upb openflow switch. https://github.com/pc2/NetFPGA-10G-UPB-OpenFlow
5. OMNeT++ Discrete Event Simulator. https://omnetpp.org/
6. Ryu sdn framework. http://osrg.github.io/ryu/
7. Introducing ONOS: A SDN network operating system for service providers. Whitepaper, Open Networking Lab ON.Lab (2014). URL http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf

8. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A Berkeley view of Cloud computing. Tech. Report UCB/EECS-2009-28, Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, USA (2009)
9. Assuncao, M.D., Carpa, R., Lefevre, L., Gluck, O.: On designing sdn services for energy-aware traffic engineering. In: 11th EAI International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom'16) (2016)
10. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer Networks **54**(15), 2787–2805 (2010)
11. Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., Xiao, X.: Overview and principles of internet traffic engineering. RFC 3272 (Informational) (2002). URL http://www.ietf.org/rfc/rfc3272.txt
12. Banikazemi, M., Olshefski, D., Shaikh, A., Tracey, J., Wang, G.: Meridian: an SDN platform for cloud network services. IEEE Communications Magazine **51**(2), 120–127 (2013)
13. Bolze, R., Cappello, F., Caron, E., Daydé, M., Desprez, F., Jeannot, E., Jégou, Y., Lantéri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Primet, P., Quetier, B., Richard, O., Talbi, E.G., Iréa, T.: Grid'5000: a large scale and highly reconfigurable experimental Grid testbed. Int. Journal of High Performance Computing Applications **20**(4), 481–494 (2006)
14. Borylo, P., Lason, A., Rzasa, J., Szymanski, A., Jajszczyk, A.: Anycast Routing for Carbon Footprint Reduction in WDM Hybrid Power Networks with Data Centers. In: IEEE Int. Conf. on Communications (ICC 2014), pp. 3714–3720. IEEE (2014)
15. Borylo, P., Lason, A., Rzasa, J., Szymanski, A., Jajszczyk, A.: Fitting Green Anycast Strategies to Cloud Services in WDM Hybrid Power Networks. In: IEEE Global Communications Conference (GLOBECOM 2014), pp. 2592–2598. IEEE (2014)
16. Borylo, P., Lason, A., Rzasa, J., Szymanski, A., Jajszczyk, A.: Energy-Aware Fog and Cloud Interplay Supported by Wide Area Software Defined Networking. In: IEEE Int. Conf. on Communications (ICC 2016). IEEE, Kuala Lumpur, Malaysia (2016)
17. Borylo, P., Lason, A., Rzasa, J., Szymanski, A., Jajszczyk, A.: Green Cloud Provisioning Throughout Cooperation of a WDM Wide Area Network and a Hybrid Power IT Infrastructure. Journal of Grid Computing (Springer) **14**(1), 127–151 (2016)
18. Carpa, R., Dias de Assuncao, M., Glück, O., Lefevre, L., Mignot, J.C.: Responsive Algorithms for Handling Load Surges and Switching Links On in Green Networks. In: IEEE Int. Conf. on Communications (ICC 2016). Kuala Lumpur, Malaysia (2016)
19. Carpa, R., Gluck, O., Lefevre, L.: Segment routing based traffic engineering for energy efficient backbone networks. In: IEEE Int. Conf. on Advanced Networks and Telecommunications Systems (ANTS 2014), pp. 1–6 (2014)
20. Develder, C., Leenheer, M.D., Dhoedt, B., Pickavet, M., Colle, D., Turck, F.D., Demeester, P.: Optical Networks for Grid and Cloud Computing Applications. Proceedings of the IEEE **100**(5), 1149–1167 (2012)
21. Goiri, I., Katsak, W., Le, K., Nguyen, T.D., Bianchini, R.: Designing and managing data centers powered by renewable energy. IEEE Micro **34**(3), 8–16 (2014)
22. Gunaratne, C., Christensen, K., Nordman, B., Suen, S.: Reducing the energy consumption of ethernet with adaptive link rate (ALR). IEEE Transactions on Computers **57**(4), 448–461 (2008)
23. Gupta, M., Singh, S.: Greening of the internet. In: ACM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '03, pp. 19–26. ACM, New York, USA (2003)
24. He, K., Khalid, J., Gember-Jacobson, A., Das, S., Prakash, C., Akella, A., Li, L.E., Thottan, M.: Measuring control plane latency in sdn-enabled switches. In: ACM SIGCOMM Symposium on Software Defined Networking Research, SOSR '15, pp. 25:1–25:6. ACM, New York, USA (2015)
25. Jeanvoine, E., Sarzyniec, L., Nussbaum, L.: Kadeploy3: Efficient and Scalable Operating System Provisioning. USENIX ;login: **38**(1), 38–44 (2013)
26. Kilper, D.: Energy challenges in access and aggregation networks. Symposium on Communication Networks Beyond the Capacity Crunch (2015). URL https://royalsociety.org/events/2015/05/communication-networks/
27. Kim, J., Cha, B., Kim, J., Kim, N.L., Noh, G., Jang, Y., An, H.G., Park, H., Hong, J., Jang, D., Ko, T., Song, W.C., Min, S., Lee, J., Kim, B., Cho, I., Kim, H.S., Kang, S.M.: Proceedings of the asia-pacific advanced network. OF@TEIN: An OpenFlow-enabled SDN Testbed over International SmartX Rack Sites **36**, 17–22 (2013)
28. Kurek, T., Niemiec, M., Lason, A.: Taking back control of privacy: a novel framework for preserving cloud-based firewall policy confidentiality. International Journal of Information Security pp. 1–16 (2015)
29. Melazzi, N., Detti, A., Mazza, G., Morabito, G., Salsano, S., Veltri, L.: An openflow-based testbed for information centric networking. In: Future Network Mobile Summit (FutureNetw 2012), pp. 1–9 (2012)
30. Miyazaki, T., Popescuy, I., Chino, M., Wang, X., Ashizawa, K., Okamotoz, S., Veeraraghavan, M., Yamanaka, N.: High speed 100GE adaptive link rate switching for energy consumption reduction. In: Int. Conf. on Optical Network Design and Modeling (ONDM 2015), pp. 227–232 (2015)
31. Nedevschi, S., Popa, L., Iannaccone, G., Ratnasamy, S., Wetherall, D.: Reducing network energy consumption via sleeping and rate-adaptation. In: 5th USENIX Symposium on Networked Systems Design and Implementation, NSDI'08, pp. 323–336. USENIX Association, Berkeley, USA (2008)
32. Ooteghem, J.V., Taylor, S., Grace, P., Lobillo, F., Smirnov, M., Demeester, P.: Sustaining a federation of future internet experimental facilities. Tech. Rep. 101436, Int. Telecommunications Society (ITS) (2014)
33. Peterson, L., Muir, S., Roscoe, T., Klingaman, A.: PlanetLab architecture: An overview. Tech. Rep. PDN-06-031, PlanetLab Consortium, Princeton, USA (2006)
34. Pfaff, B., Pettit, J., Koponen, T., Jackson, E.J., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., Casado, M.: The design and implementation of open vSwitch. In: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2015) (2015)
35. Rossigneux, F., Gelas, J.P., Lefevre, L., de Assuncao, M.D.: A generic and extensible framework for monitoring energy consumption of OpenStack clouds. In: SustainCom 2014, pp. 696–702 (2014)
36. Ryu project team: RYU SDN Framework - English Edition. Ryu SDN Framework Community (2014)

37. 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: T. Korakis, M. Zink, M. Ott (eds.) Testbeds and Research Infrastructure, Development of Networks and Communities, *LNICST*, vol. 44, pp. 372–372. Springer Berlin Heidelberg (2012)
38. Sarolahti, P., Kuznetsov, A.: Congestion control in linux tcp. In: USENIX Annual Technical Conference, FREENIX Track, pp. 49–62 (2002)
39. Szymanski, A., Lason, A., Rzasa, J., Jajszczyk, A.: Performance evaluation of the grade-of-service-based routing strategies for optical networks. In: IEEE International Conference on Communications (ICC 2008), pp. 5252–5257. IEEE (2008)
40. Thanh, N.H., Nam, P.N., Truong, T.H., Hung, N.T., Doanh, L.K., Pries, R.: Enabling experiments for energy-efficient data center networks on OpenFlow-based platform. In: Fourth International Conference on Communications and Electronics (ICCE 2012), pp. 239–244 (2012)
41. Vasić, N., Kostić, D.: Energy-aware traffic engineering. In: 1st Int. Conf. on Energy-Efficient Computing and Networking, e-Energy '10, pp. 169–178. ACM, New York, USA (2010)
42. Vu, T.H., Nam, P.N., Thanh, T., Hung, L.T., Van, L.A., Linh, N.D., Thien, T.D., Thanh, N.H.: Power aware OpenFlow switch extension for energy saving in data centers. In: International Conference on Advanced Technologies for Communications (ATC 2012), pp. 309–313 (2012)
43. Zhang, M., Yi, C., Liu, B., Zhang, B.: GreenTE: Power-aware traffic engineering. In: 18th IEEE Int. Conf. on Network Protocols (ICNP 2010), pp. 21–30 (2010)