An SDN-Based Energy-Aware Traffic Management Mechanism

Alex B. Vieira
Universidade Federal de Juiz de Fora
Wallace Nascimento Paraizo
Universidade Federal de Juiz de Fora
Luciano Jerez Chaves
Universidade Federal de Juiz de Fora
Luiz H. A. Correa
Universidade Federal de Lavras
Edelberto Franco Silva (edelberto@ice.ufjf.br)
Universidade Federal de Juiz de Fora https://orcid.org/0000-0002-0058-9260

Research Article

Keywords: Green computing, Power consumption, Software Defined Networking, Traffic management

Posted Date: April 15th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-23017/v1

Version of Record: A version of this preprint was published at Annals of Telecommunications on June 30th, 2021. See the published version at https://doi.org/10.1007/s12243-021-00863-x.

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

An SDN-Based Energy-Aware Traffic Management Mechanism

Alex B. Vieira · Wallace Nascimento Paraizo · Luciano Jerez Chaves · Luiz H. A. Correia · Edelberto Franco Silva

Received: February 13th, 2020 / Accepted: date

Abstract Green computing is a central theme in many computer science areas, including computer networks. Dynamic solutions that can properly adjust network resources can prevent infrastructure over-provision and mitigate power consumption during low demand periods. In this work, we propose DTM (Dynamic mechanism for Traffic Management), an energy-aware dynamic mechanism for traffic management, built upon the SDN paradigm. DTM continuously monitors the use of network links to concentrate traffic and disconnect idle equipment without degrading the offered quality of service. Our simulations show that the mechanism can save up to 46% of energy on average. When compared to an existing solution, DTM is, on average, 4% to 7% better, depending on the evaluated scenario.

Keywords Green computing, Power consumption, Software Defined Networking, Traffic management.

1 Introduction

Energy-aware computing is a central theme of research today, especially in computer networks. Resources are often overprovisioned to cope with demands and spike in network traffic, resulting in wasted resources and high power consumption. Part of this problem occurs due to the existing infrastructure of legacy networks, where stability is crucial to business success. In these networks, the control and routing are inseparable and embedded in network el-

Alex B. Vieira, Wallace Nascimento Paraizo, Luciano Jerez Chaves and Edelberto Franco Silva
Computer Science Department, Universidade Federal de Juiz de Fora, Brazil
E-mail: {alex.borges,wallace.paraizo}@ufjf.edu.br; {luciano.chaves, edelberto}@ice.ufjf.br

Luiz H. A. Correia
Computer Science Department, Universidade Federal de Lavras, Brazil
E-mail: lcorreia@ufla.br

ements. In this sense, the adoption of new technologies and documents in production environments is virtually unviable [1].

The new Software Defined Networks (SDN) paradigm makes a clear separation between data and control planes. A centralized view and the network programmability allows one to easily monitor network and to program its flow routing. SDN can assist in shutting down (or hibernating) network devices as a way to save power.

Note that there are several ways to save power in networks such as optimizing the network infrastructure, the dynamic switch on-off of network elements, the scaling of the frequency of network elements, and the controlling of the link capacity (i.e., data rate). However, support for some of these approaches is not common for commercial network devices. For example, scaling the frequency of network elements or controlling the link capacity was initially part of the IEEE 802.3az standard. However, due to the implementation challenges, the functionality for configuring Ethernet network elements in different data rates has been excluded from this standard final specification [2]. In this sense, most of the existing solutions to save power on networks rely on the optimizing of data routing.

In this paper, we present DTM: an Energy-Aware Dynamic Traffic Management. DTM employs SDN to improve traffic routing between switches. Briefly, DTM monitors network flows and uses fewer routes as possible to transmit the existing end-to-end flows, which enables the disconnection of links and the turning off of idle network devices. Moreover, to avoid the recurrent connection/disconnection of devices —which may shorten equipment life— our mechanism uses a traffic aggregation policy. However, a policy only aggregates flows, allowing link disconnection, case its utilization level does not indicate a future increasing demand.

We emphasize that the existing works in the literature focus on topology optimization and the minimization of the number of links and switches. Existing solutions do not consider the dynamics of flows [3–6]. In fact, only few works like [7] and [5] explore the seasonality of traffic. Even so, their solutions are sometimes computationally costly.

We have evaluated DTM by emulating an SDN environment from a typical university campus topology (section 5). We consider different levels of traffic as well as distinct overloaded scenarios. Our evaluation results evidence an energy-saving from 17% to 46%, depending on the emulated network demand. Compared with an existing solution, our results are, on average, 7% to 4% better for medium and high demand scenarios, respectively.

The remainder of this article is organized as follows. In section 2, we discuss the related work. In section 3, we introduce the common scenario we consider in this work. In Section 4 we describe the protocol we propose. Evaluation and results of our proposal are presented in Section 5. Finally, Section 6, concludes this work.

2 Related Work

Traditionally, energy-saving works on computer networks have focused on mathematical models of topology and minimizing attributes (*e.g.* number of links, switches, or CPU frequency) [3–6]. The authors of [8] formulate an optimization model to minimize the number of network links and network assets, choosing the position of the controller. According to them, these are key issues for energy saving. Many of these formulations involve heuristics, once resource optimization in this environment is considered an NP-Complete problem. The authors of [4] and [6] propose heuristics for disconnecting aggregated link network cards, which can be disabled independently. The authors do not turn off the entire equipment or an entire link because they believe this reduces network connectivity. In [9], the authors consider bandwidth, link load, and traffic arrays to hibernate switches, rearranging the network topology as needed.

[8] formulates an optimization model that considers the routing requirements for data and control planes. This model considers the reduction of network assets and also considers driver location, which, according to the authors, is a key issue for energy savings.

These problem formulations often involve heuristics, because resource allocation optimization in this environment is an NP-Complete problem. [4] and [6] propose heuristics to reduce power consumption by disconnecting aggregated link network cards, which can be disabled independently. The authors do not turn off equipment or entire links because they believe this reduces network connectivity. [9] investigates the reduction of network infrastructure power consumption through the hibernation of switches, rearranging the network topology as needed. For this, they consider bandwidth, link load and traffic arrays as input parameters for their decision algorithms.

Rather than rearranging the network topology or shutting down the networking assets, the authors of [3] work by adjusting the speed of underused links. The lower priority traffic is redirected and real-time traffic is kept on the minimum path to satisfy the desired QoS. Note that the reduction in power consumption is achieved only by reducing link speeds, resulting in marginal gains. Moreover, also note that link speed reduction is not commonly supported on commercial devices.

Only a few works consider the dynamics of the network. Clearly, nowadays, network dynamics and infrastructure is a topic that must be taken into account for many purposes, from energy-saving to the coexistence of multiple high-performance distributed applications [10]. In this context, Heller et al. propose the *ElasticTree*, a network power manager for *datacenter* that monitors traffic and chooses the set of network elements that must remain active to meet the goals of performance and fault tolerance. The *ElasticTree* turns off unnecessary links and switches as much as possible. For this, a formal model, a greedy algorithm, a topology-sensitive heuristic, and forecasting methods were adopted. In the same way, the authors of [5] present a model and a greedy heuristic to find a minimal path concerning the number of hops and an

increase in energy consumption. Thus, the authors consider that other network elements can be turned off.

Recently, a number of works use SDN to save energy in networks, especially, in cloud-based environments. Son et al. [11] also avoid overbooking of resources, and as a consequence, save energy, through the use of SDN. Authors, in this case, strategically allocate a more precise amount of resources to VMs and traffic (in SDN-Based Cloud Data Centers) according to the demand of Quality of Service (QoS). Despite the similar objective (the energy-saving), DTM focuses on network resources and topology, while Son et al. focus on the allocation of Cloud Data Centers resources. In a different way, Xu et al. [12] schedule the flow order to save energy in an SDN network compliant environment. In this way, they can optimize the selection of links, avoiding defragmentation. Finally, Jia et al. [13] also use an SDN approach to re-route flows. Initially, they treat the problem as a min-cost problem, which is NP-hard. Moreover, they do not dynamically define network link utilization levels, which may induce network to shift between on-off link states, in a ping-pong effect. Finally, we highlight that we provide close-to-real implementation, instead of simulation.

Differently from DTM, most existing work does not consider the network dynamism or only present costly solutions that rely on specialized mechanisms. Moreover, these studies only reassess the proposed optimization/heuristic in the face of changes on flows but do not consider a smoothing in interference with network assets or possible traffic aggregations. Thus, they only disconnect links and switches when they become naturally idle.

3 Considered Scenario

In this article, we consider networks with an arbitrary topology as shown in Figure 1. At the network edge, host nodes represent the elements that generate and consume data streams. Hosts are connected to access switches, which in turn are connected to routing switches. All switches are compatible with the OpenFlow protocol, which is the most widely used SDN platform today, both in development and research [1]. The switches are responsible for routing packets between links, according to the flow rules previously configured by one or more controllers, responsible for the centralized control logic of the network.

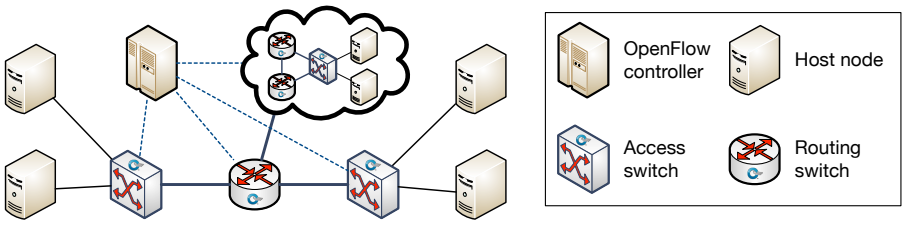


Fig. 1 Considered arbitrary network topology.

This network topology is modeled as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \mathcal{N}_h \cup \mathcal{N}_c$ is a set of nodes, and $\mathcal{L}$ a set of links. The subset $\mathcal{N}_h$ represents the hosts, as a subset $\mathcal{N}_c$ corresponding to the swtiches. Specifically, $\mathcal{N}_c = \mathcal{N}_a \cup \mathcal{N}_e$, where $\mathcal{N}_a$ is the access switches, and $\exists\, l_{ci} \in \mathcal{L}$ connects a switch c directly attached to a host $i \in \mathcal{N}_h$; and $\mathcal{N}_e$ the forward switches, where $\forall\, l_{cj} \in \mathcal{L}$ connect the switch c directly to another switch $j \in \mathcal{N}_c$.

DTM reduces the power consumption by shutting down (or hibernating) selectively ports on switches that have their idle links. If all ports on a switch are turned off, the switch can also be turned off. For this, traffic management algorithms need to concentrate flows on a minimum number of links, but without degrading the quality of service offered. This strategy is called resource consolidation, according to [14].

For all switch $c \in \mathcal{N}_c$, denote $E_t(c)$ as the energy consumption of c at time t, following the equation 1 [3]:

$$E_t(c) = Eb_t(c) + Ep_g(c) \times Np_{gt}(c) + Ep_f(c) \times Np_{ft}(c) \tag{1}$$

where $Eb_t(c)$ shows the base consumption of c at time t, necessary to keep the switch working (processor, cooler, etc.); $Ep_g(c)$ and $Ep_f(c)$ show the energy consumption for each port of 1 Gbps and 100 Mbps in c, respectively; $Np_{gt}(c)$ and $Np_{ft}(c)$ represent the number of 1 Gbps e 100 Mbps ports active in c at time t, respectively. Note that the energy consumption of each port changes according to its speed [5]. Then, the energy consumption can be estimated for network total amount $E_t(\mathcal{G})$ at time t following the equation 2:

$$E_t(\mathcal{G}) = \sum_{c \in \mathcal{N}_c} E_t(c) \tag{2}$$

4 DTM: A Dynamic Traffic Management

In this article, we propose an SDN-based energy-aware traffic management mechanism. This Dynamic Traffic Management engine (DTM) presents three main components: (i) *active network monitoring*, which maintains an up-to-date network resource usage model; (ii) the *new flows installation* algorithm, which reactively identifies new traffic and allocates appropriate routes; and (iii) the *active stream redirection* algorithm, which aggregates flows in the least amount of links to shut down idle resources, avoiding to overload the remaining links.

In what follows, we detail each one of these components.

4.1 Active Network Monitoring

DTM uses the OpenFlow controller to access network information. In this sense, the network controller maintains a detailed and up-to-date network view, including information on resource use and energy consumption of active elements. Since DTM can shut down and reconnect links and switches over

time, we denote as $\mathcal{G}'_t \subseteq \mathcal{G}$ nodes and links set that is on and available for use in an instant t. To keep the model up to date, the controller periodically sends OpenFlow messages to the switches requesting information and statistics. The Δt interval between requests can be adjusted to a good compromise between model update and network overhead. With the answers, it is possible to update $\mathcal{G}'_t$, the estimated power consumption $E_t(\mathcal{G})$, the instantaneous throughput of data streams and also the utilization rate of network links. This information is posteriorly used by the new flow installation algorithm and active flow redirection algorithm.

Let $U_t(l)$ be the link utilization rate $l \in \mathcal{L}$, calculated by the bandwidth ratio at the instant t for the transmission capacity of l. In this work, we use Table 1 as a reference to classify the link utilization and any number of link states could be used. This categorization is used by algorithms to prevent overloaded links from receiving new flows, avoiding packet loss. Besides, low-load links prove to be favorable candidates for shutdown ports on switches. Thus, the mechanism aims to reallocate all flows of these links to alternative routes, making them idle and saving energy. We let as future work the investigation of the impact of varying the possible number of link states.

Table 1 $S_t(l)$ state as a function of $U_t(l)$ use.

$S_t(l)$	Link State	Utilization Rate $U_t(l)$
s_0	low load	$0\% \leq U_t(l) < T_{0\to1}$
s_1	normal load	$T_{0\to1} \leq U_t(l) < T_{1\to2}$
s_2	high load	$T_{1\to2} \leq U_t(l) < T_{2\to3}$
s_3	overload	$T_{2\to3} \leq U_t(l) \leq 100\%$

Once the controller collects information of network elements (links) and updates the network model, it builds the $\mathcal{P}_{ij}$ set with the k minimum paths connecting host i to j, $\forall\, i, j \in \mathcal{N}_h$. The controller also calculates, $\forall\, c \in \mathcal{N}_c$, the $Q(c)$ indicator representing the total paths in the $\mathcal{P}$ sets passing the c switch. Minimum paths and the $Q(c)$ indicator make sense when viewed from an energy-saving perspective because the shorter and more concentrated the paths, the fewer network elements are used and the more switches can be turned off.

4.2 Flow Installation

The new flow installation by the controller is reactive. In other words, when a host 'a' starts a new flow to host 'b', the network controller detects this new flow routes accordingly.

Let f_{ab} be a flow between hosts $a, b \in \mathcal{N}_h$. When the first packet of f_{ab} reaches the access switch, the controller is notified and starts searching for the best path $p_{ab}^{best} \in \mathcal{P}_{ab}$ with all links in state $S_t(l) \leq s_{max}$, as per Algorithm 1. The initial search is restricted to paths that do not use overloaded

links ($s_{max} = s_2$). The algorithm prioritizes the optimal path p_i that can be accommodated in $\mathcal{G}'_t$ and does not result in increased power consumption. In the case of several ideal paths, the shortest one is prioritized $|p_i|$, having as a tiebreaker the indicator $Q(p_i) = \sum_{c \in p_i} Q(c)$. If there is no $p_i \in \mathcal{G}'_t$, then the energy impact $E_i(p)$, $\forall p \in \mathcal{P}_{ab}$, and choose the p_e path that results in the smallest final energy increment. In this case, will be need to update $\mathcal{G}'_t$, enabling links or switches to accommodate the new flow.

Algorithm 1: PATHFINDER

```
Input: P_ab ways; Active subgraph G'_t; s_max state.
Output: best path p_ab^best ∈ P_ab with S_t(l) ≤ s_max links.
p_i ← ∞;   p_e ← ∞;
foreach p ∈ P_ab do
    if ∀l ∈ p / S_t(l) ≤ s_max then
        if p ∈ G'_t then
            if (|p| < |p_i|) OR (|p| = |p_i| e Q(p) > Q(p_i)) then
                p_i ← p;
        if E_i(p) < E_i(p_e) then
            p_e ← p;
if p_i ≠ ∞ then return p_i else return p_e;
```

In the case it does not exist a path p^m_{ab} with at least one overloaded link, the controller relax the maximum state of links constraint ($s_{max} = s_3$), and perform a new search for p^m_{ab}. Once the path is set to f_{ab}, flow rules are installed on all $c \in p^m_{ab}$ switches.

4.3 Active Flows Redirection

The new flow installation considers the instantaneous links utilization rate. However, traffic behavior over time can lead to undesirable use in some links (i.e., link overloading). In this case, the active flow redirection algorithm redistributes traffic across the network. To do this, it evaluates the $S_t(l)$ state that each link can assume, as shown in Figure 2 (which reflects the states we previously defined in Table 1). In this work, we assume that flows allocated to a link in an *overload* state $S_t(l) = s_3$ may be penalized with longer delays and packet losses. In this case, the algorithm redirects from this to other links the smallest amount of flows required for the utilization rate to decrease to some state $S_t(l) < s_3$. This strategy prevents links from remaining overloaded, reducing possible damage to network QoS indicators.

On the other hand, we also consider that links at the *low charge* state $S_t(l) = s_0$ are underutilized. In this case, the algorithm tries to redirect all of this flows to other links in the network an then, disable this link which in turn saves network energy. When a link is completely idle it is possible to shut

down its ports on the adjacent switches. If all ports on a switch are off then the switch can also be turned off completely. If there is no way able to receive any of the redirecting flows, the process is aborted with the understanding that it is not possible to vacate the link without damaging the QoS indicators of the active traffic.

Flows redirection is performed as a trigger whenever active network monitoring classifies a link as low load or overload. The search for alternative paths is done by the Algorithm 1, with the restriction of $s_{max} = s_1$. This restriction requires new paths to use only links in the normal or low load state, preventing links in the s_2 (high load) state receiving more flows and being classified as overloaded in sequence, which would result in new redirections and, in a subsequent "ping-pong" effect between the s_2 and s_3 states.

5 Evaluation and Results

In this section, we evaluate the DTM and the energy it saves. More precisely, we evaluate DTM in three distinct scenarios. First, we evaluate it in a static scenario (Section 5.1), where we illustrate the very basic functionality of DTM. Then, we evaluate DTM in a dynamic, yet synthetic, scenario (Section 5.2), where we show the performance of DTM under a controlled experiment. Finally, we evaluate DTM under a realistic scenario (Section 5.3), where we mimic a typical campus network and evaluate the performance of our proposal.

We have evaluated DTM using the network emulator *Mininet* [1] (ver. 2.3.0), which allows the creation of a virtual host network, switches, links, and controllers in a single computer. The OpenFlow driver used was POX (ver. 0.2.0) and the switches were run on instances of *OpenVSwitch* [2] (ver. 2.5. 2). The experiments were performed on a computer with 7th generation Intel Core i5@3.1GHz, 8GB RAM and Lubuntu 16.04 operating system.

[1] `http://mininet.org`
[2] `https://www.openvswitch.org/`

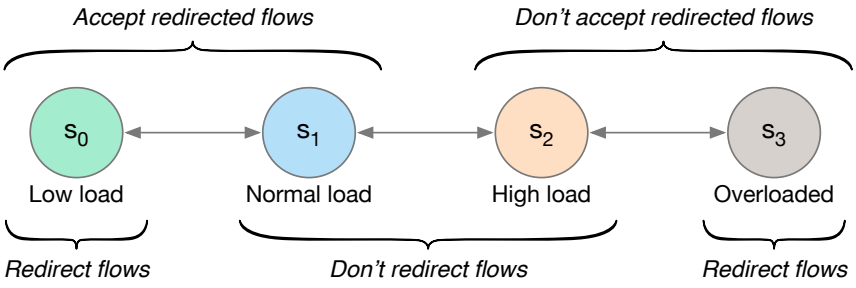
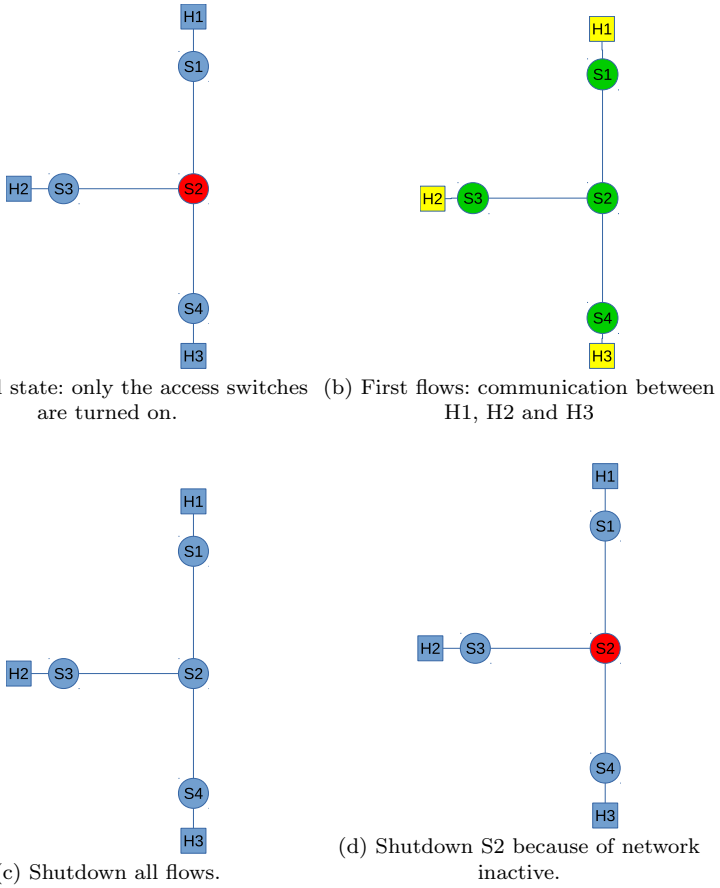


Fig. 2 $S_t(l)$ states assumed by a l link.

5.1 Scenario A - Static Evaluation of DTM

First, we statically evaluate DTM to check its internal mechanism accuracy. This illustrative example guides readers to follow the basic DTM mechanisms. In a static built scenario, we are able to test (i) the DTM network topology discovery process; (ii) the active network monitoring process; (iii) the flow installation process.



(a) Initial state: only the access switches are turned on.

(b) First flows: communication between H1, H2 and H3

(c) Shutdown all flows.

(d) Shutdown S2 because of network inactive.

Fig. 3 Simple static evaluation of DTM.

Figure 3 presents the static scenario we use to check DTM basic functionality. According to this figure, the topology we evaluate presents three hosts (h1, h2, h3) and for switches (S1, S2, S3).

First, the DTM controller performs a network topology discovery process. It detects all switches and links. Initially, only access switches will be kept turned on (S1, S3, and S4), and all reminders switches will be turned off to

save energy (s2). Note that in Figure 3(a) S2 is red-colored to represent it in off state.

After a few seconds, as shown in Figure 3(b), we intentionally start a flow between hosts H1, H2, and H3. The network controller then detects these flows and starts the flow installation process. In this case, the controller must activate the switches S2, otherwise, there will no path between communicating endpoints. In sum, the active nodes and links are colored in green in Figure 3(b).

Flows cease after 10 seconds. In this case, hosts and links present no inactivity, as we depict in Figure 3(c). In this experiment, we set up a two-minute threshold that triggers the DTM network controller. The controller, in turn, turns-off all unnecessary switches, saving as much as energy as it could. By the end, Figure 3(d) presents the same network state as the initial setup.

5.2 Scenario B - Dynamic evaluation of DTM

Second, we dynamically evaluate DTM by changing the average network load in a controlled network environment. In this scenario, we evaluate the energy-savings, as well as the overhead DTM, imposes to the network due to flow redirections.

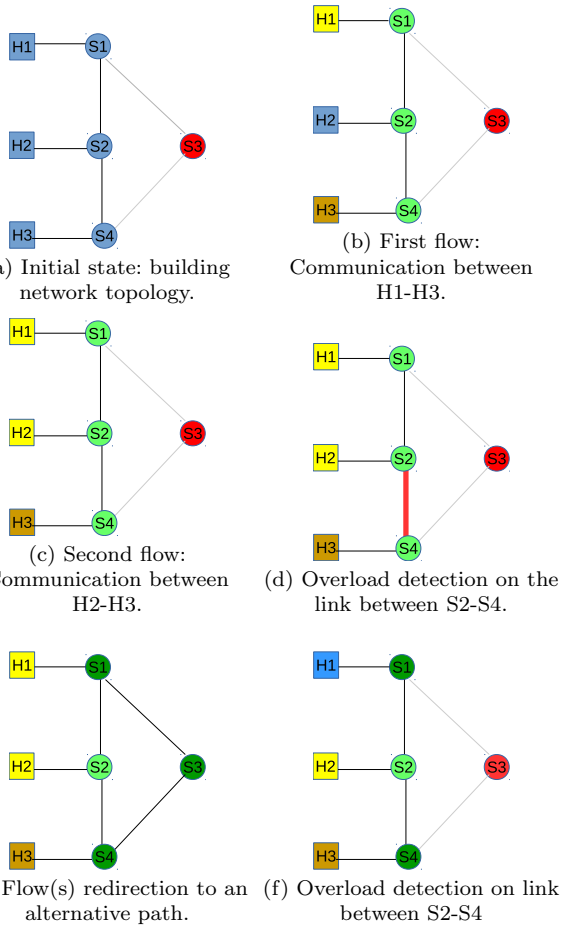
Figure 4 illustrates the network we evaluate. The switches are represented by circles and the hosts by squares. All links are Gigabit Ethernet. Initially, as shown in Figure 4(a), end hosts are idle and, as a consequence, switches are in the low load state we previously defined. The controller initiates the automatic network topology detection and the, it builds the minimum spanning tree which connects all hosts. In this case, note, in Figure 4(a), that switch S3 is off, which saves network energy.

During our experiments, the host H3 acts as a server. Host H1 connects to H3, demanding on average 500Mbps. The controller calculates the minimum spanning tree and, as shown in Figure 4(b), the suggested shortest path between H1 and H3 is S1-S2-S4. In this case, switches S1,S2 and, S4 are low loaded and they accept the flow. The controller then finally installs the route in these switches tables.

In a third step, shown in Figure 4(c), the host H2 initiates communication with the server H3. H2 demands, in this case, 300Mbps. In short, the controller identifies the path S2-S4 in the minimum spanning tree. These switches (i.e., S2 and S4) are in the normal load state and then, they accept the flow; the route is installed and hosts communicate with each other.

Any network link may become overloaded. For example, as shown in Figure 4(d), the link between S2 and S4 turns overloaded which imposes the controller to redirect traffic, according to the policies we previously defined.

In short, the controller performs an *active flows redirection* policy and it notes that the flow between H1 and H3 has an optional route through S1-S3-S4. The switch S3 can handle the overloaded traffic from the other switches. The controller then binds and installs this route, as shown in Figure 4(e).



(a) Initial state: building network topology.

(b) First flow: Communication between H1-H3.

(c) Second flow: Communication between H2-H3.

(d) Overload detection on the link between S2-S4.

(e) Flow(s) redirection to an alternative path.

(f) Overload detection on link between S2-S4

Fig. 4 Behavior of the model proposed.

At this moment, there is no one overloaded switch, however, all switches are turned on, which imposes the highest energy consumption to the network.

Suppose flow between H1 and H3 ends. In this case, switch S3 (and its links) will turn to the low load state. The DTM controller may redirect all remaining flows from S3 table to other network switches (which are on normal or low load states). As a consequence, case the controller successfully transfers all S3 flows to other switches, it turns off S3, as shown in Figure 4(f)).

Figure 5 shows the percentage of energy-saving, and the network load, during the experiment we previously depicted. As expected, the energy economy is close-related to the network load. In fact, the higher the network load, the higher the number of switches we expect to turn on. Note that the flow redirection mechanism may impose a latency on energy-savings. For example, during the period from 17s to 21s in Figure 5, the DTM controller turned on switches due to overloaded links, which impacted the network energy consump-

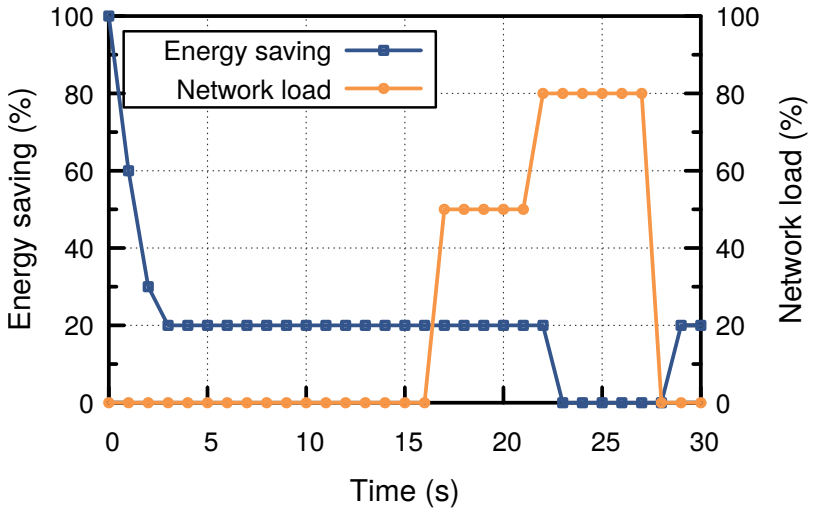


Fig. 5 Energy-saving versus network load.

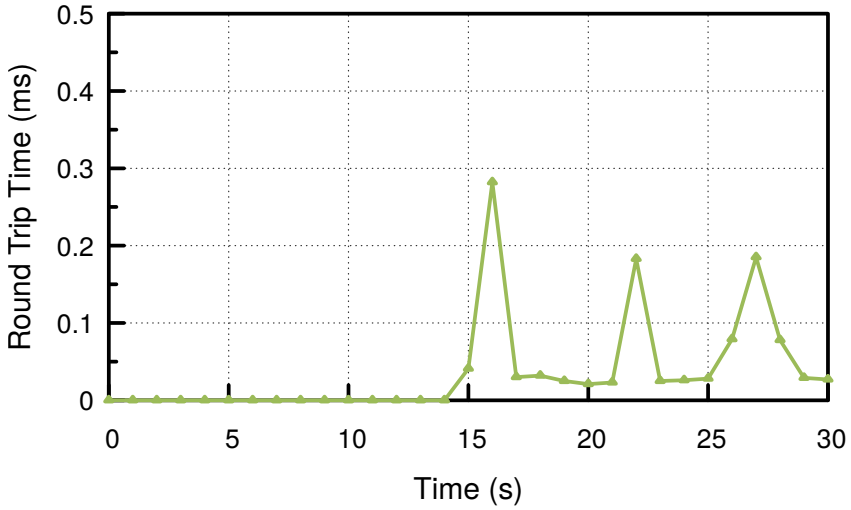


Fig. 6 Latency at each moment of the simulation.

tion. During our experiments, the DTM controller reaction time was about 0.2 milliseconds.

Flow installation process and flow redirections may injure network traffic. For example, during a flow redirection, end-to-end communication may experience higher latencies. Figure 6 presents the end-to-end latency between network end-points during our experiments. This figure clear corroborates our previous comment. In special, during flow installation/redirections, as we no-

ticed around the 15 to the 28 seconds of the experiment. More precisely, during a flow installation, a switch must consult the controller for a forwarding rule (reactive mode). Controller answer switch and, only after that, switch installs the new rule and flow can be performed. The second peak corresponds to a flow redirection due to an overloaded link. As the controller had proactively installed the rules, the overhead is smoothly lower. The third peak is similar to the second. That is, the controller observes that there is a low load on the routing switch links, triggering an event that leads to a new stream redirection section, intending to turn off the switch.

5.3 Scenario C - Realistic Evaluation of DTM

DTM can be used in arbitrary topologies as long as the nodes are SDN compatible. In this paper, we consider a realistic scenario represented by Figure 7, which shows a campus network topology. This kind of topology has already been largely explored in similar works, for example, [5].

In this work, we consider a network topology with 95 links and 45 nodes (i.e., switches). More in deep, nodes include routing switches (#1 through #4) and access switches (#5 through #18). The others are host nodes (#19 to #45), forming two distinct groups (the upper and the lower portion of the figure). Client/server application pairs for traffic generation and consumption are always positioned one in each host group.

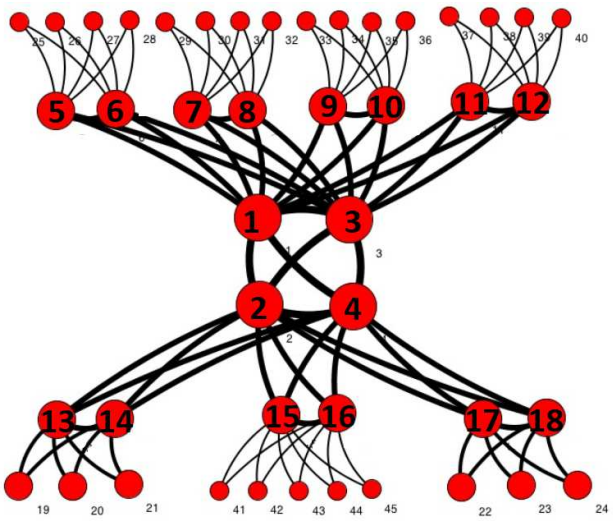


Fig. 7 University Campus Like Network Topology.

We evaluate DTM considering two distinct scenarios. First, we consider the network with homogeneous links. In this case, all links have the same

negotiated speed of 250Mbps, which is close to the previous works[5]. This allows us to compare DTM to an existing approach to save energy in networks. Then, we evaluate DTM considering heterogeneous network links. In this case, the links interconnecting the routing switches have a speed of 1Gbps, while the other links maintain the speed of 250Mbps. Again, this scenario follows close to previous existing work.

5.3.1 Evaluation Methodology

In this article, we consider the power consumption parameters (in Watts) as the same as [3] and [15]. More specifically, the base switch consumption is $Eb_t(c) = 146$, the 1 Gbps port consumption is $Ep_g(c) = 0.87$ and the 100 Mbps port consumption is $Ep_f(c) = 0.18$. New flows arrival rate follows a Poisson process with an interval expectation $\lambda = 3$ sec. New flows have a fixed duration of 15 sec. and can assume one of the load levels described in the table 2. These values are equivalent to those used in [5] and take into account night traffic behavior, average daytime traffic, and annual peak traffic. According to the authors, annual peak traffic is five times higher than night traffic, and average daytime traffic is three times higher than night traffic. The traffic was generated by the D-ITG (Internet Traffic Generator) [16] tool.

Table 2 Load levels according to traffic patterns.

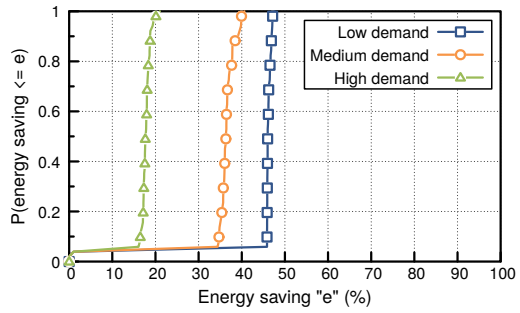
Level	Traffic demand	Description	flow throughput
N_1	High	Annual peak	$]150, 250]$ Mbps
N_2	Average	Daytime average	$]50, 150]$ Mbps
N_3	Low	Nightly average	$[0, 50]$ Mbps

The $T_{0\to1}$, $T_{1\to2}$ and $T_{2\to3}$ thresholds used to classify link states were set at 20%, 60% and 80% respectively. These values define a good compromise between the states we define in this work. We let as future work further investigation of these parameters. Moreover, we consider $\Delta t = 1$ sec. In a real environment, we must finetune this value to not overload the network with control messages. We consider building $\mathcal{P}_{ij}$ sets with $k = 16$ minimum paths connecting host i to j, $\forall\, i, j \in \mathcal{N}_h$. All minimum paths will have five hops between source and destination in this topology.

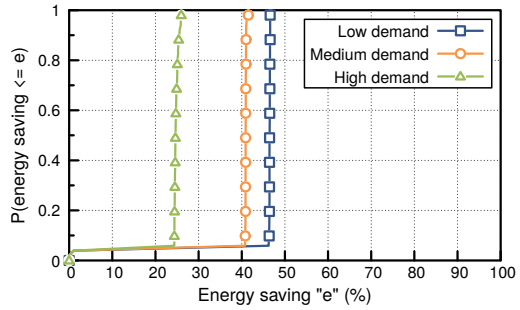
Our evaluations evidence the percentage of power savings by comparing DTM to a network where all devices are turned on (i.e., there is no energy-saving mechanism). We repeat each experiement 50 times, and each on remains for 120 seconds. Unless we tell otherwise, we present mean values and confidence intervals, for a 95% confidence level.

5.3.2 Evaluation

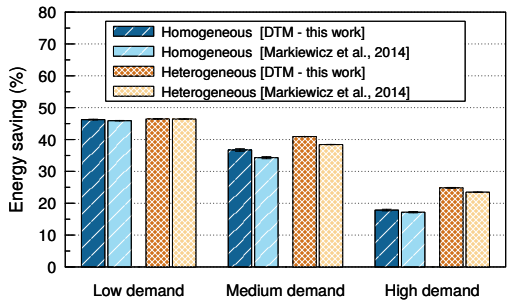
We evaluated DTM in the homogeneous and heterogeneous scenarios, considering the three traffic demand levels. Figures 8(a) and 8(b) show the energy-



(a) Energy saving in homogeneous scenario



(b) Energy saving in heterogeneous scenario

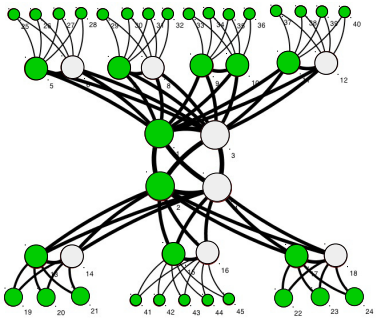


(c) DTM Comparison with state of the art

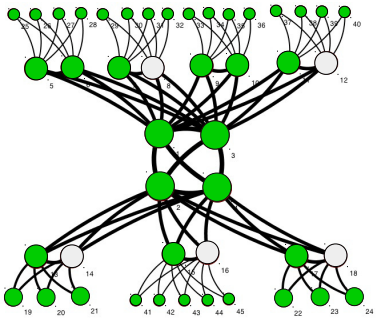
Fig. 8 Energy savings achieved in the evaluated scenarios and comparison of DTM with state of the art.

saving cumulative distribution functions in each evaluated configuration. Intuitively, the higher demand, the lower savings achieved, as more links are in use and fewer opportunities to shut down idle network elements.

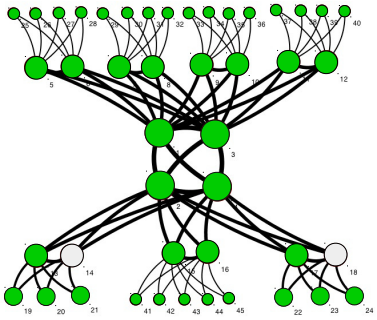
Under low traffic demand, both scenarios show similar behavior. In these cases, the minimum links required to maintain network connectivity is active. As a result, DTM achieves a significant average energy savings of over 46%. On average demand, energy savings differ. In a heterogeneous scenario, the



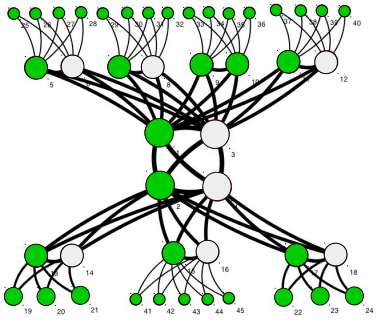
(a) Homogeneous - Low demand



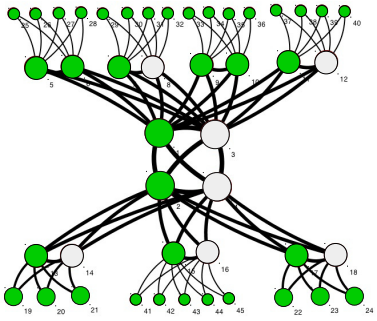
(b) Homogeneous - Medium demand



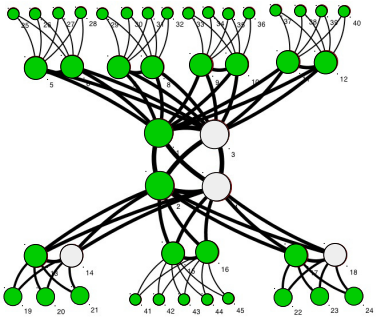
(c) Homogeneous - High demand



(d) Heterogeneous - Low demand



(e) Heterogeneous - Medium demand



(f) Heterogeneous - High demand

Fig. 9 Active switches during experiments

average savings were approximately 40%. In a homogeneous scenario, this value is 3.75% lower. As the heterogeneous scenario has some higher capacity links, it can absorb this average demand without linking new links. In the homogeneous scenario, however, it is necessary to connect more links and eventually more switches, which increases energy consumption. This is even more evident with a high work demand, where more links and switches are required to support full traffic, minimizing the opportunity to save energy.

Energy savings fall considerably for both cases but are most evident in the homogeneous scenario. In this situation, the homogeneous scenario consumed 6.58% more energy than the heterogeneous one, and the latter achieved no more than 25% of the total average energy savings.

Figure 9 presents the network map and illustrates which switches are turned on/off. In this figure, we present in green the switches typically turned on during our simulations. The figures 9 (a) to (c) present situations of low, medium and high network demands, respectively, for the homogeneous scenario. The figures 9 (d) to (f) refer to the heterogeneous scenario.

As discussed earlier, in a high demand situation, there are more switches turned on for both scenarios. On the other hand, in a low demand scenario, the flow between the network links is low, which eliminates the redirection of active flows and results in the highest energy savings.

The heterogeneous scenario has greater energy savings compared to the homogeneous one. This is quite evident when comparing the figures 9 (b) and 9 (e), or the figures 9 (c) and 9 (f). Again, heterogeneous scenarios have some links with larger capacity that are capable of supporting large volumes of flows without having to enable alternate paths for possible redirections.

Table 3 shows the flow observed in the network during simulations. Homogeneous and heterogeneous scenarios present equivalent behavior for low and medium demands. For high demand, the flow observed in the heterogeneous scenario is higher. This is because the homogeneous scenario core links reach their load limit and become a bottleneck for the network. In contrast, the heterogeneous scenario core links support the generated flows, providing energy savings and higher flow between switches.

Table 3 Confidence interval for network throughput.

Level	Demand	Homogeneous	Heterogeneous
N_1	High	[115,2; 130,3] Mbps	[135,3; 152,7] Mbps
N_2	Medium	[87,5; 98,8] Mbps	[89,9; 101,4] Mbps
N_3	Low	[23,2; 26,3] Mbps	[24,2; 27,3] Mbps

Finally, we compare DTM with a state of the art solution, specifically [5]. The figure 8(c) shows that both algorithms have the same energy-saving gain in low traffic demand scenarios. In fact, the minimum number of links for network connectivity is capable of ensuring all low demand experience flows. Thus, there are scenarios, some solutions are at their maximum level of the economy. On average traffic demand, note a single difference between the algorithms for both homogeneous and heterogeneous links. DTM was, on average, 7% more economical than the proposal of [5] in a homogeneous scenario, and 6.5%, on average, more economical in the heterogeneous scenario. A higher traffic demand turns more difficult for any mechanism to save energy, regardless of the energy-saving strategy. In this case, most of the available links must be turned on to accommodate all existing flows.

When traffic demand is high, DTM was 3.96% and 5.79% better than [5] method, considering the homogeneous and heterogeneous scenarios, respectively. In short, even when energy-saving opportunities are rare, DTM has the advantage. The gains are significant, especially considering the ultimate impact given computer network scale usage today.

6 Conclusions and Future Work

In this paper, we present DTM, an Energy-Aware Dynamic Traffic Management. DTM employs SDN to improve traffic routing between switches. The mechanism has been evaluated through the emulation of networks using Mininet, considering a realistic topology. Our evaluation results show that DTM provided an average energy saving of 46.01% in a scenario with low network demand, similar to a nighttime pattern. In this low demand scenario, as expected, we achieved the highest energy savings. In scenarios with average to high traffic demands, the mean energy saving is 36.72% and 17.86%, respectively. Compared to a well-known existing mechanism, DTM is, on average, 7% to 4% better for medium and high demand scenarios, respectively. Future work includes the investigation of DTM scalability in other topologies, ranging the topology density, for example.

References

1. Leonardo C. Costa, Alex B. Vieira, Erik de Britto e Silva, Daniel F. Macedo, Geraldo Gomes, Luiz H. A. Correia, and Luiz F. M. Vieira. Performance evaluation of OpenFlow data planes. In *IFIP/IEEE IM)*, pages 470–475, 2017.
2. B. Addis, A. Capone, G. Carello, L.G. Gianoli, and B. Sansò. Energy management in communication networks: a journey through modeling and optimization glasses. *Computer Communications*, 91–92:76–94, 2016.
3. Shingo Sasaki, Kanayo Ogura, Bhed Bahadur Bista, and Toyoo Takata. A proposal of QoS-aware power saving scheme for SDN-based networks. In *NBiS*, pages 405–410, 2015.
4. Will Fisher, Martin Suchara, and Jennifer Rexford. Greening backbone networks: reducing energy consumption by shutting off cables in bundled links. In *ACM SIGCOMM Workshops*, pages 29–34, 2010.
5. Adam Markiewicz, Phuong Nga Tran, and Andreas Timm-Giel. Energy consumption optimization for software defined networks considering dynamic traffic. In *IEEE CloudNet*, pages 155–160, 2014.
6. Gongqi Lin, Sieteng Soh, Kwan-Wu Chin, and Mihai Lazarescu. Efficient heuristics for energy-aware routing in networks bundled links. *Computer Networks*, 57(8):1774–1788, 2013.
7. Brandon Heller, Srini Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree: Saving energy in data center networks. In *USENIX NSDI*, pages 1–17, 2010.
8. Adriana Fernández-Fernández, Cristina Cervelló-Pastor, and Leonardo Ochoa-Aday. Improved energy-aware routing algorithm in software-defined networks. In *IEEE LCN*, pages 196–199, 2016.
9. Khan Mohammad Habibullah, Eric Rondeau, and Jean-Philippe Georges. Reducing energy consumption of network infrastructure using spectral approach. In M. Dastbaz, H. Arabnia, and B. Akhgar, editors, *Technology for Smart Futures*, pages 235–250. Springer, Cham, 2018.

10. Alexandre T Oliveira, Bruno José CA Martins, Marcelo F Moreno, Antônio Tadeu A Gomes, Artur Ziviani, and Alex Borges Vieira. SDN-based architecture for providing quality of service to high-performance distributed applications. *International Journal of Network Management*, page e2078, 2019.
11. Jungmin Son, Amir Vahid Dastjerdi, Rodrigo N Calheiros, and Rajkumar Buyya. Sla-aware and energy-efficient dynamic overbooking in sdn-based cloud data centers. *IEEE Transactions on Sustainable Computing*, 2(2):76–89, 2017.
12. Guan Xu, Bin Dai, Benxiong Huang, Jun Yang, and Sheng Wen. Bandwidth-aware energy efficient flow scheduling with sdn in data center networks. *Future Generation computer systems*, 68:163–174, 2017.
13. Xuya Jia, Yong Jiang, Zehua Guo, Gengbiao Shen, and Lei Wang. Intelligent path control for energy-saving in hybrid sdn networks. *Computer Networks*, 131:65–76, 2018.
14. Aruna Prem Bianzino, Claude Chaudet, Dario Rossi, and Jean-Louis Rougier. A survey of green networking research. *IEEE Communications Surveys & Tutorials*, 14(1):3 – 20, 2010.
15. Priya Mahadevan, Puneet Sharma, Sujata Banerjee, and Parthasarathy Ranganathan. Energy aware network operations. In *IEEE INFOCOM*, pages 25–30, 2009.
16. Alessio Botta, Alberto Dainotti, and Antonio Pescapè. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, 2012.

Figures

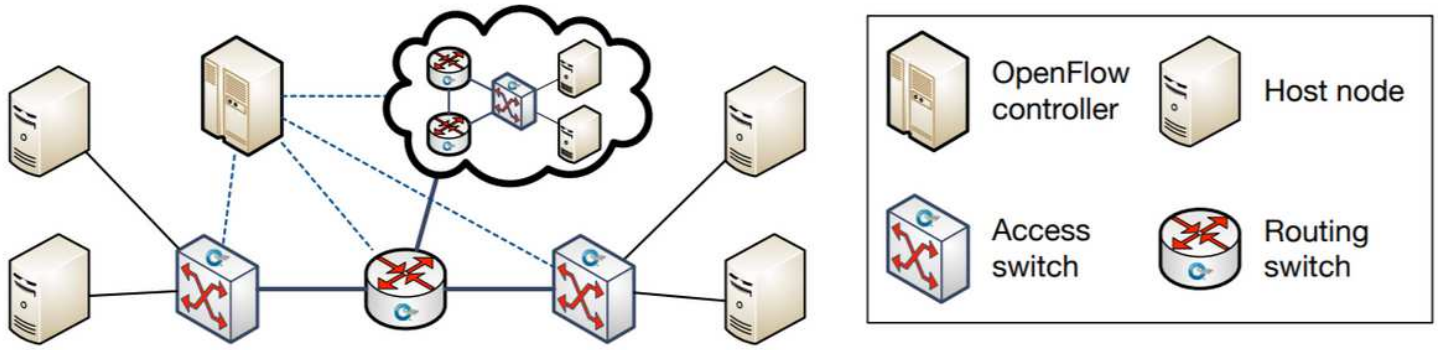


Figure 1

Considered arbitrary network topology.

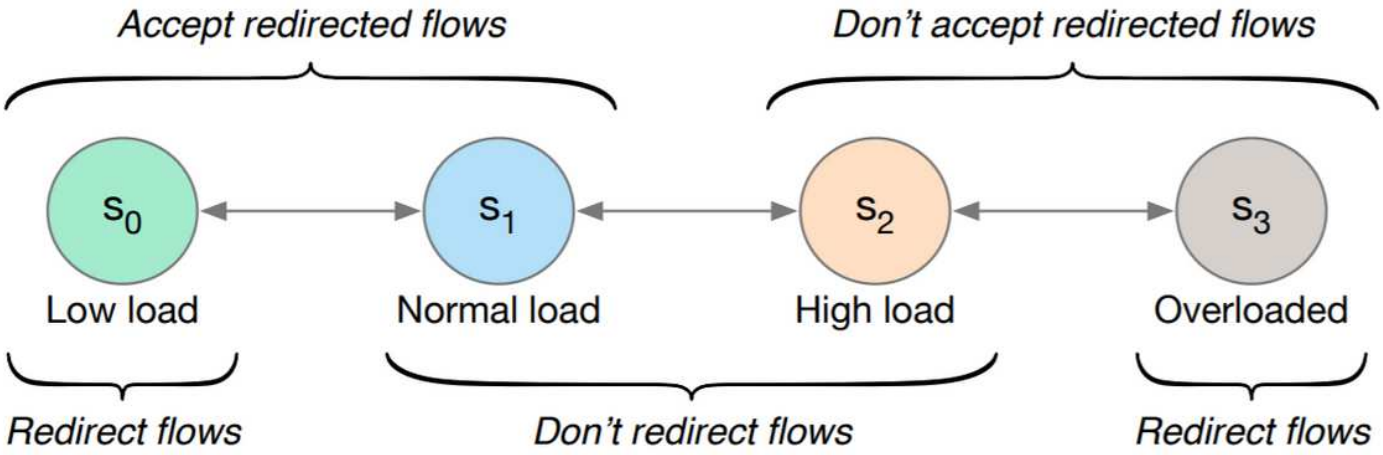
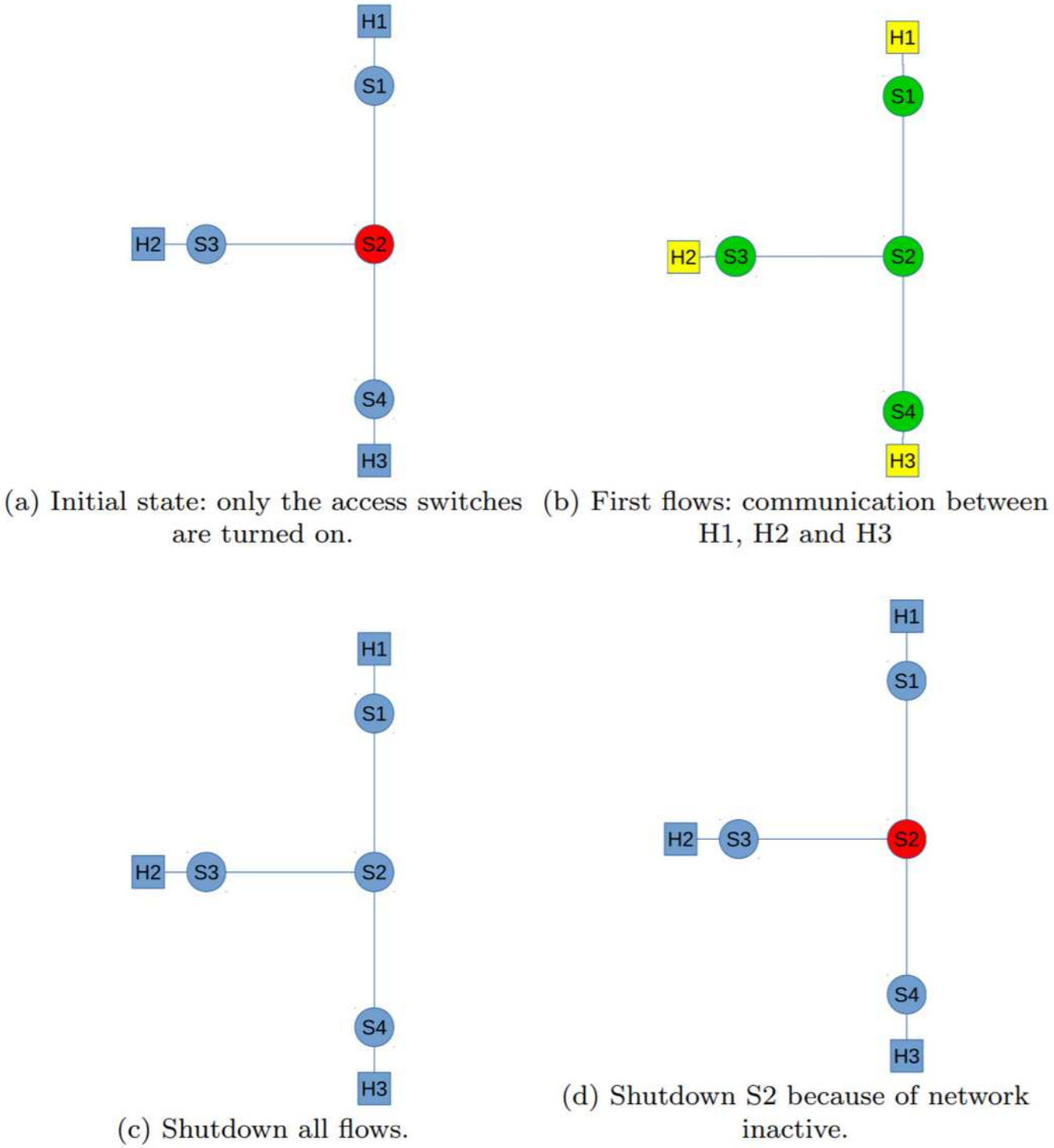


Figure 2

St(l) states assumed by a l link.



(a) Initial state: only the access switches are turned on.

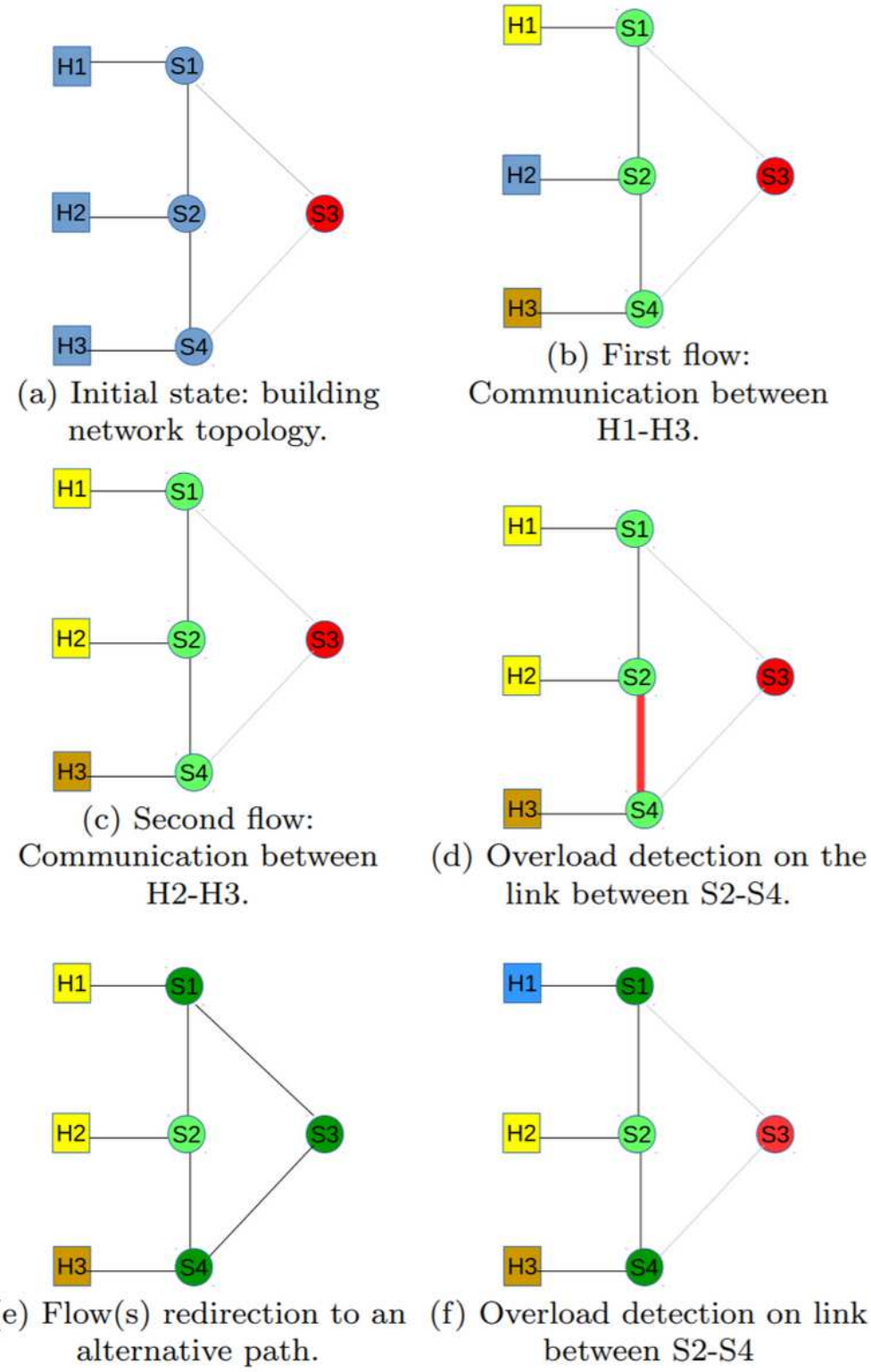
(b) First flows: communication between H1, H2 and H3

(c) Shutdown all flows.

(d) Shutdown S2 because of network inactive.

Figure 3

Simple static evaluation of DTM.



(a) Initial state: building network topology.

(b) First flow: Communication between H1-H3.

(c) Second flow: Communication between H2-H3.

(d) Overload detection on the link between S2-S4.

(e) Flow(s) redirection to an alternative path.

(f) Overload detection on link between S2-S4

Figure 4

Behavior of the model proposed.

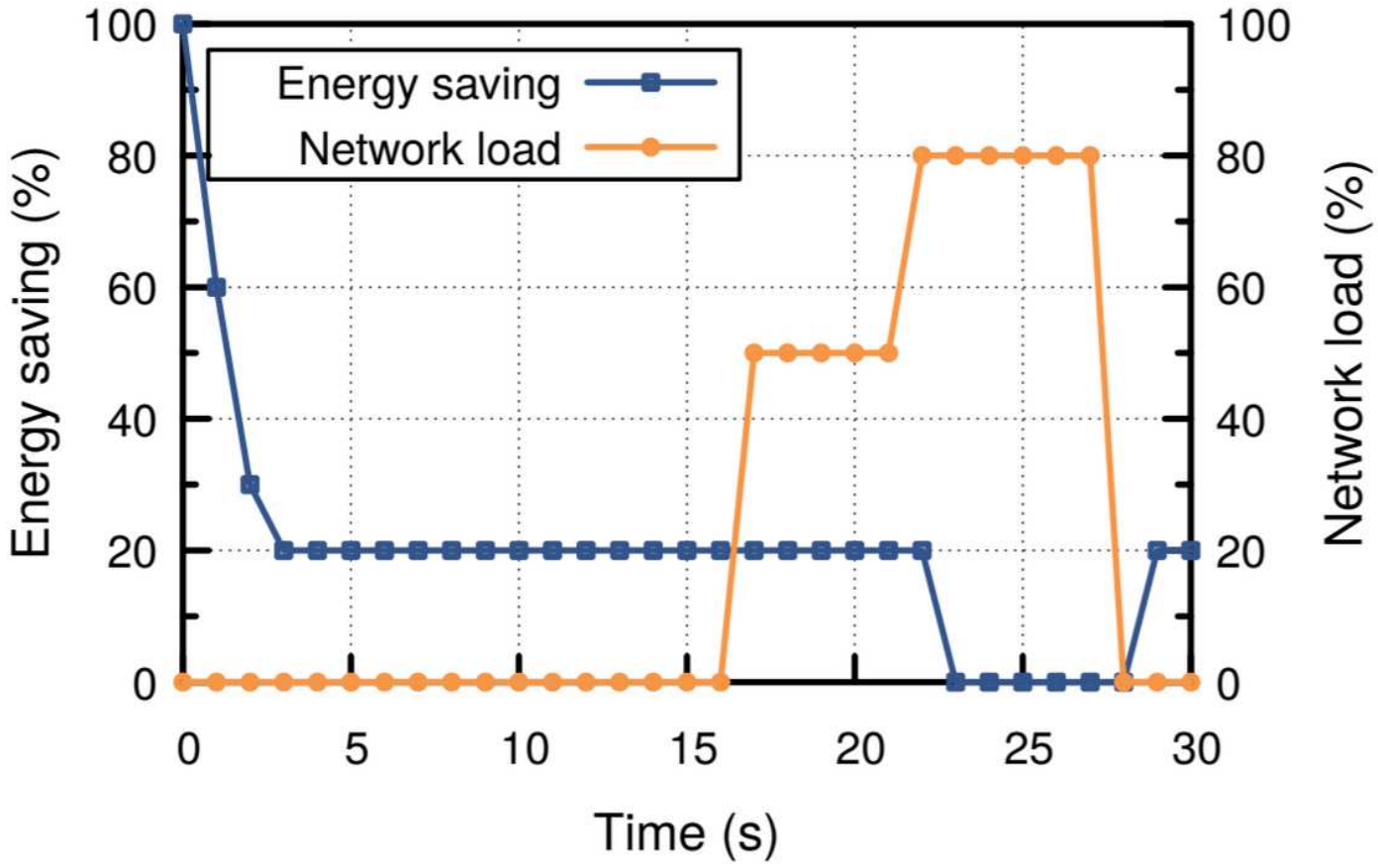


Figure 5

Energy-saving versus network load.

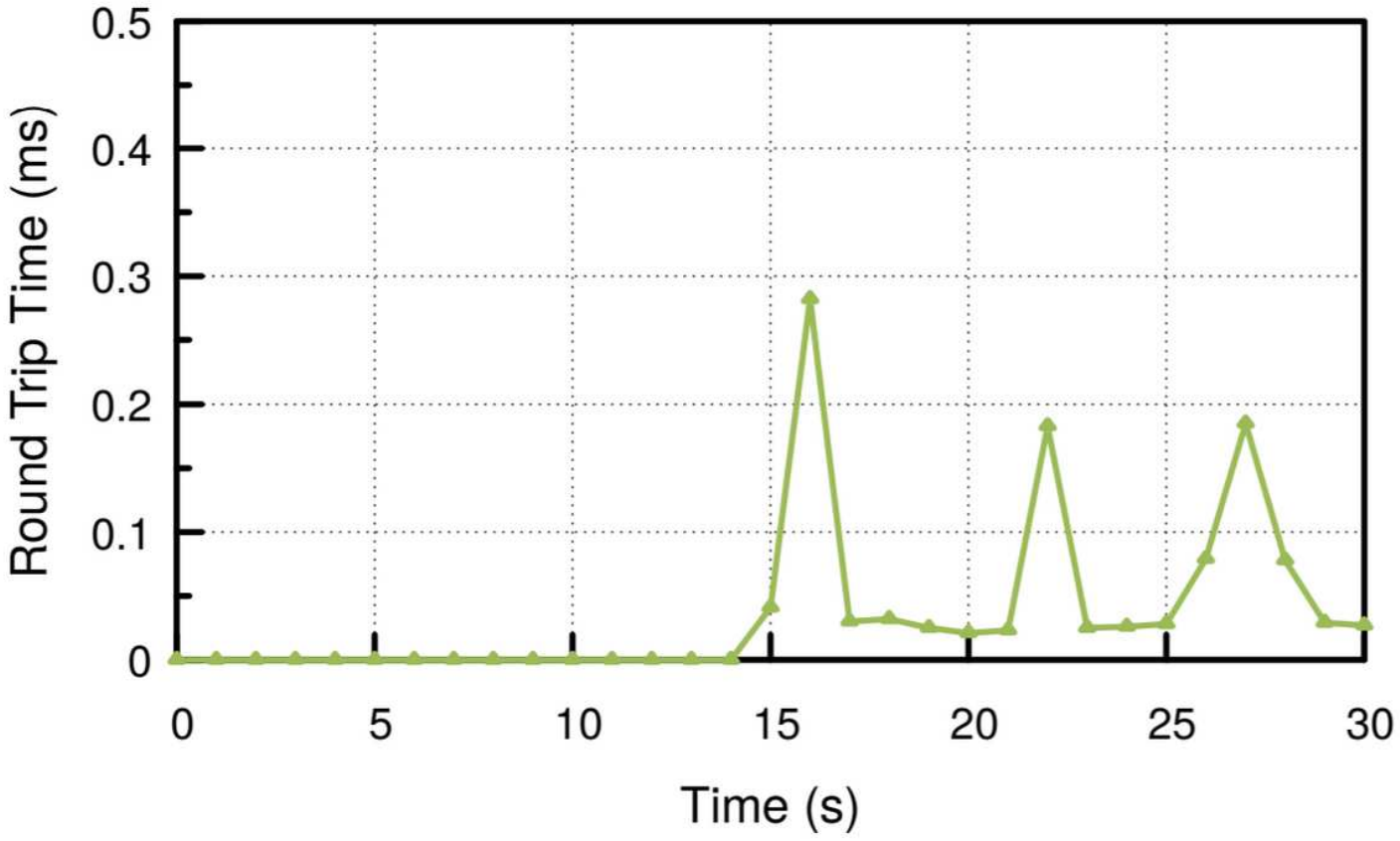


Figure 6

Latency at each moment of the simulation.

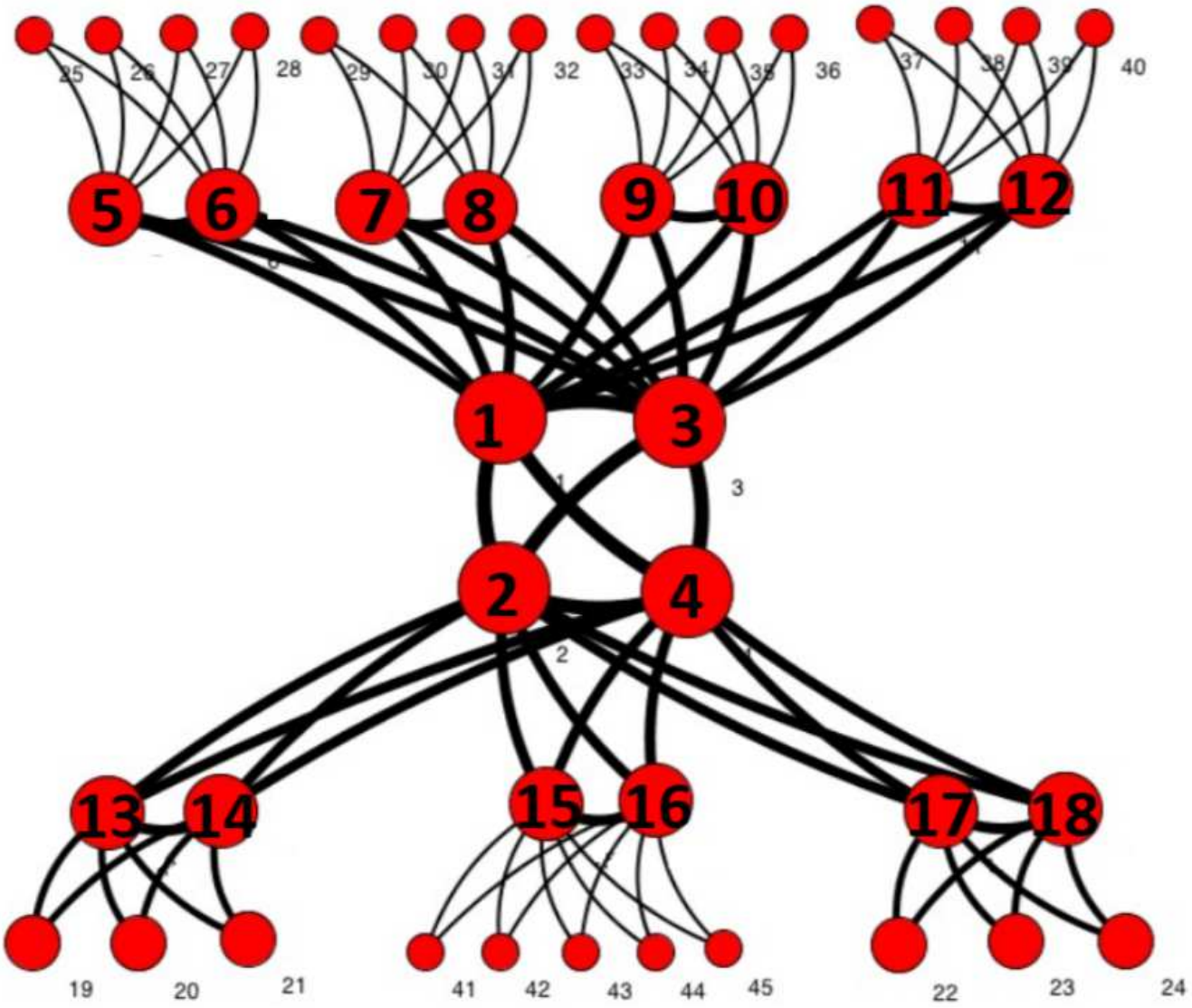
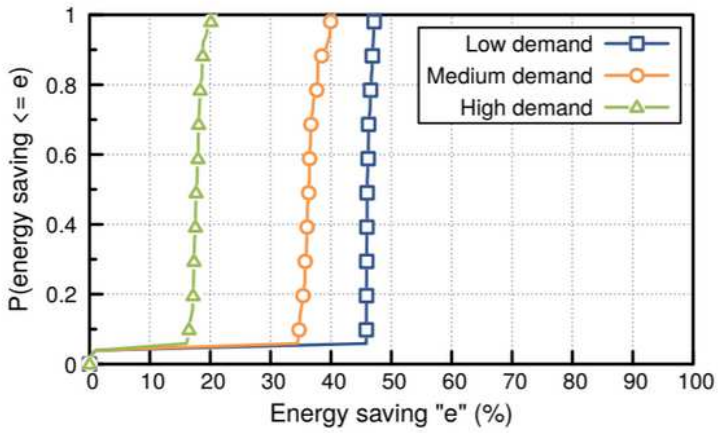
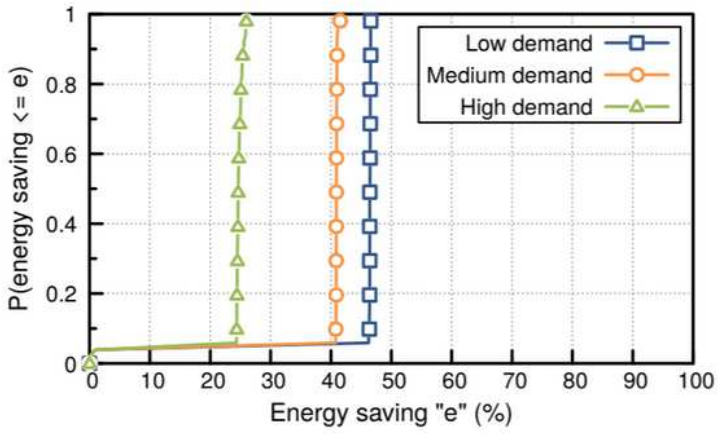


Figure 7

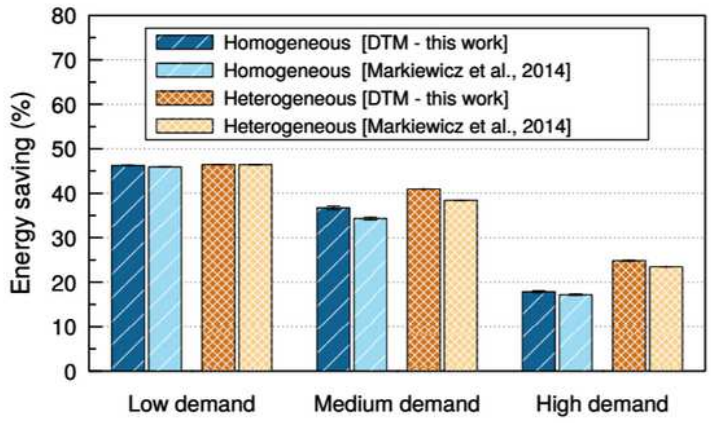
University Campus Like Network Topology.



(a) Energy saving in homogeneous scenario



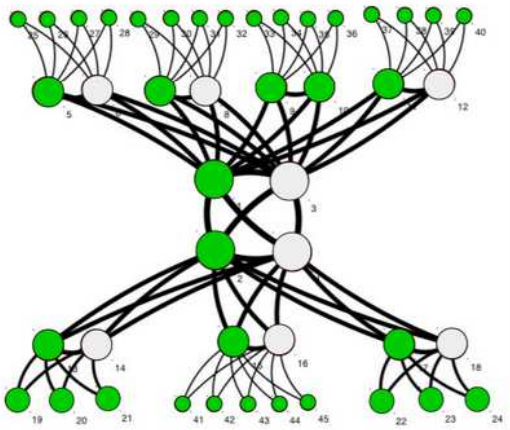
(b) Energy saving in heterogeneous scenario



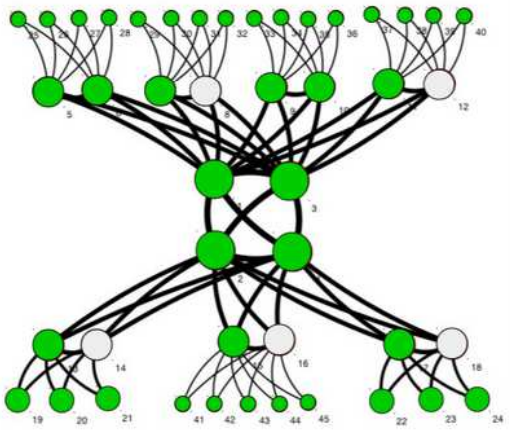
(c) DTM Comparison with state of the art

Figure 8

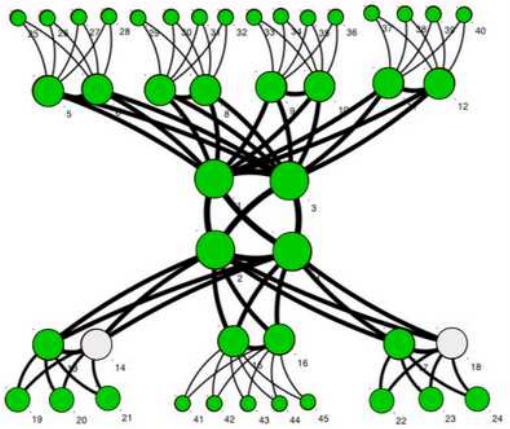
Energy savings achieved in the evaluated scenarios and comparison of DTM with state of the art.



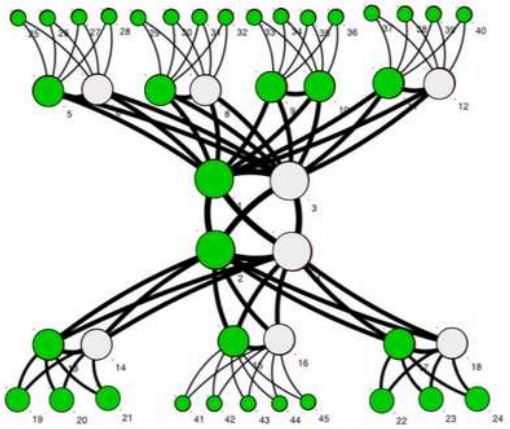
(a) Homogeneous - Low demand



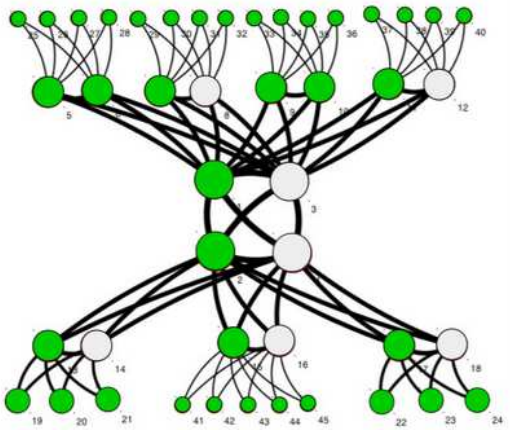
(b) Homogeneous - Medium demand



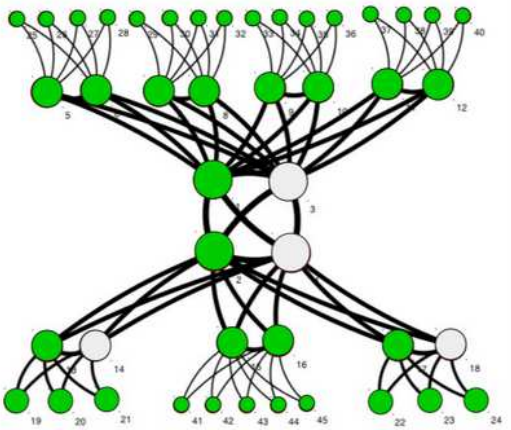
(c) Homogeneous - High demand



(d) Heterogeneous - Low demand



(e) Heterogeneous - Medium demand



(f) Heterogeneous - High demand

Figure 9

Active switches during experiments.