# GPF: A GREEN POWER FORWARDING TECHNIQUE FOR ENERGY-EFFICIENT NETWORK OPERATIONS

Rahil Gandotra[1] and Levi Perigo[2]

[1]Interdisciplinary Telecom Program, University of Colorado Boulder, USA
[2]Department of Computer Science, University of Colorado Boulder, USA

***ABSTRACT***

*The energy consumption of network infrastructures is increasing; therefore, research efforts designed to diminish this growing carbon footprint are necessary. Building on prior work, which determined a difference in the energy consumption of network hardware based on their forwarding configurations and developed a real-time network energy monitoring tool, this research proposes a novel technique to incorporate individual device energy efficiency into network routing decisions. A new routing metric and algorithm are presented to select the lowest-power, least-congested paths between destinations, known as Green Power Forwarding (GPF). In addition, a network dial is developed to enhance GPF by allowing network administrators to tune the network to optimally operate between energy savings and network performance. To ensure the scope of this research for industry adoption, implementation details for different generations of networking infrastructure (past, present, and future) are also discussed. The experiment results indicate that significant energy and, in turn, cost savings can be achieved by employing the proposed GPF technique without a reduction in network performance. The future directions for this research include developing dynamically-tuning network dial modes and extending the principles to inter-domain routing.*

***KEYWORDS***

*Energy efficiency, intent-based networking, network optimization dial, SDN, programmable control plane, OpenFlow, programmable data plane, P4.*

## 1. INTRODUCTION

The energy consumption of the information and communications technology (ICT) infrastructure has been rising steadily - from 1815 TWh or 8% of the global energy use in 2012 to 2,547 TWh or 10% of the global energy use in 2017 [1]. These high energy consumptions translate into large carbon footprints since coal, the most carbon-intensive approach to producing energy, still remains the single largest source of power generation globally (36.4%) [2]. Researchers have predicted the global energy demand of the ICT sector by 2030 to be 8,265 TWh or 21% of the global energy use, with the worst-case prediction being 30,715 TWh or 51% of global energy use [3]. The network infrastructure alone is predicted to consume 2,641 TWh by 2030 making it a considerable portion of the total ICT energy consumption (32%). Therefore, research efforts to attempt to reduce it are warranted.

While there have been significant improvements in the energy efficiency of data centers and networks, they have mostly been overwhelmed by the growth of data traffic caused by increased

David C. Wyld et al. (Eds): MLDS, NECO, SEMIT, IBCOM, SPPR, SCAI, CSIA, ICCSEA - 2021
pp. 83-97, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.111808

consumption of services [4, 5]. For instance, Van Heddeghem et al. estimated that the growth of electricity consumed by networks (at 10.4% per year) was much faster than the global electricity demand itself (at 3% per year) [6]. While compute equipment is designed to be reasonably energy proportional, i.e. their energy consumption is proportional to their load, networking equipment is not energy proportional because it is a shared resource and is expected to always be available [7]. The numerous efforts put into compute energy management and cooling technologies have resulted in networks becoming a bigger fraction of the total ICT power budget [8].

This paper builds on prior research conducted on comparing the energy efficiencies of networking hardware [9] and developing a real-time network power monitoring framework [10], with the objective to incorporate network energy efficiency into the network-wide forwarding decisions, and to study the tradeoffs between energy savings and network performance. [9] presented a detailed energy efficiency study of a variety of network devices and analyzed the power consumption when traffic flows through individual devices as a function of their configuration. The test results utilized an in-line power meter and indicated that significant differences in energy efficiency of devices were prevalent based on their forwarding configurations, such as the number or type of the forwarding entries, or whether the forwarding entries were stored in the hardware or the software table. In order to enable the non-intrusive collection of power information from network devices, [10] presented a framework employing information models - both standardized IETF and non-standard customized models – to demonstrate the feasibility of real-time data gathering for effective network energy management.

The motivation behind the research in this study is to build on previous work by developing energy-efficient forwarding mechanisms that incorporate individual device efficiencies into network-wide forwarding decisions with minimal impact to network performance. The results from this work could be used to enhance the body of knowledge in the vital field of network energy management. The rest of the paper is organized as follows: Section 2 discusses related work and highlights the novelty of this work; Section 3 describes the research methodology – both the algorithm and its different mechanisms; Section 4 discusses the research results; Section 5 present the conclusions.

## 2. RELATED WORK

The existing work on incorporating energy awareness into networks can be categorized as five distinct techniques: sleeping of network elements, link rate adaptation, proxying, store and forward, and network traffic aggregation.

Sleeping of network elements was based on the intuition that because computer networks are provisioned for peak load and are generally underutilized, parts of the network or individual devices can be put to sleep during off-peak hours [11]. The work involved in this approach was two-fold: predicting low-utilization periods suitable for energy savings, and reconfiguring the network to put network elements to sleep [10]. The target network element to be put to sleep could either be the physical device(s) [13-15], or individual components of one device [16-19]. Since networking devices typically provide limited support for putting the system or components to sleep, the techniques in this category face the challenges of waking up a sleeping element and understanding the impact of sleeping on protocols which require network presence.

Link rate adaptation was identified as a potential approach after research showed that high link rates (1/10 Gbps) consumed more energy than low link rates (10/100 Mbps) [20]. The work in this approach involved identifying periods of low traffic wherein the links could be operated at slower rates, and reconfiguring the device link speeds network wide [21]. Both reactive approaches, based on matching the link speeds with the current utilization [22, 23], and proactive

approaches, predicting future traffic patterns from historical utilization data [17, 24], were proposed. However, since several routing and switching protocols operate based on link speed metrics, the interoperability of such protocols with the link rate adaptation techniques remains to be examined.

Proxying is based on the hypothesis that periodic network chatter traffic does not allow end hosts to sleep effectively and that network elements could be employed as proxies to enable infrastructure-level savings [20]. The techniques proposed in this category attempted to counter the requirements necessitating networked end-hosts to be connected and responsive even while sleeping, i.e. always-on should not mean always powered on [25]. Both on-system [21, 26] and off-system [27, 28] proxies were proposed to offload some of the end-host responsibilities. The impact of proxying on session-based protocols and the additional energy consumption of a proxy as compared to the amount of energy saved are some of the open problems in this category.

The observation that many web applications exhibit bursty traffic with high inter-arrival traffic times led to the development of store and forward techniques [29]. While the traditional practice in network engineering is to avoid bursts using congestion avoidance and QoS queues, the insight behind the techniques in this area was to shape traffic in bursts in order to allow network elements to be put in low power/sleep mode in between these bursts [17]. The techniques in this category included in-network bunch-and-burst [30, 31] and a parallel network implementing time-based scheduling in conjunction with the Internet [32, 33]. However, practical implementations have demonstrated these techniques to be suitable only for specific network topologies due to the cost of an added end-to-end delay and the problem of synchronizing bursts across the network to enable efficient sleeping.

Network traffic aggregation techniques aimed to proactively find the minimum-power subsets of the infrastructure that can provide the desired level of performance while enabling energy savings [23]. The work in this category involved collecting traffic statistics such as the traffic matrix and the desired fault-tolerance levels, optimizing the problem while satisfying all constraints, and reconfiguring the devices to put them in low-power/sleep modes [34, 35]. The major open problems in this category include finding methods to reduce the computing time that allows for on-the-fly solutions and the general lack of support for putting network hardware to sleep or wake them back up.

While only a few of the proposed techniques have seen industry adoption, for example link mode adaptation was standardized as IEEE 802.3az, a majority of them suffer from critical flaws. Firstly, most networking hardware deployed has limited support for putting a device or its individual components into sleep or low-power modes. Since network devices are expected to be always on and connected, the vendors typically provide minimal energy management capabilities. Secondly, waking up a sleeping element is a non-trivial problem. Unless the device is connected to a ‘smart’ networked power strip, a device which is powered off becomes laborious to connect back. Thirdly, the impact of frequent device/component-level changes on cost-based switching and routing protocols would need to be minimized. Frequent and incorrect changes could in fact lead to an increase in the energy consumption because powering on a sleeping element or transitioning to another mode consumes additional energy and time.

This paper presents a sixth distinct and novel technique for saving energy in networks – energy-efficient forwarding - by utilizing individual device energy efficiencies for the network forwarding decisions. In other words, the aim of this technique was to deliver traffic between endpoints in a network while attempting to consume the minimum amount of energy as possible. The proposed technique does not make any assumptions about the underlying hardware support for sleep/low-power modes, and instead employs the differences in energy efficiency of different

devices while processing traffic. An approach to provide tradeoffs between energy savings and network performance is also presented. The proposed technique includes both an algorithm for making forwarding decisions as well as a mechanism for implementing it over the current and the next generation of networking paradigms. Since the technique aims to be transparent to applications and to reduce operational energy consumption, it could be used in conjunction with any of the five abovementioned techniques to allow for further energy savings.

## 3. RESEARCH METHODOLOGY

The metricidentified in previous work to compare energy efficiency between devices was Energy Consumption Rating (ECR), which is the aggregated energy consumption normalized to the actual throughput. A lower metric implied the device consumed less energy while processing the same amount of traffic. While this metric is helpful in determining which device processes traffic more energy efficiently, it is not useful for making forwarding decisions and suffers from two major problems: the congestion problem, and the sleeping giant problem.

The congestion problem is presented as: consider two similar network devices A and B. If at any instance, device A is experiencing much higher amount of traffic as compared to device B, its ECR value would be much lower than device B's. However, a low value of ECR does not imply that additional traffic could be forwarded through device A, as it is already heavily congested.

The sleeping giant problem is presented as: Consider two dissimilar network devices A and B. At any instance, both devices are experiencing similar amounts of bandwidth, however, since device B is a bigger, more feature-rich box, it is consuming higher amounts of power which makes its ECR value higher than device A's. Therefore, even though device B has the potential to support higher amounts of bandwidth, it will not be preferred at this instance because of its current lower ECR value.

So, a more useful metric instead for making forwarding decisions would be one that helps in determining the lowest-power and least-congested paths. In other words, devices that consume less power and have high available bandwidth (Avail_Bandwidth) should be preferred over those consuming high power and having low available bandwidth. The metric Energy Efficiency Potential (EEP) in introduced and used in this research, which is the aggregated energy consumption normalized to the available bandwidth of the device.

The technique presented in this paper, called Green Power Forwarding (GPF), consists of both a GPF algorithm for making decisions about finding the most energy-efficient paths in the network, and mechanisms for implementing the GPF algorithm over different varieties of network infrastructures such as traditional, programmable-control, and programmable-data networks. The GPF algorithm is based on Dijkstra's Shortest Path First (SPF) algorithm but instead of using link speed as the routing metric, Watt/Avail_Mbps is used to compute least-power, least-congested paths.

Algorithm 1. GPF algorithm

**Given→**
nodes: vertices in the graph; denoted by u, v
ee: energy efficiency values of each node; denoted by $ee_N$
weights: weighted edges connecting two nodes; denoted by w(u,v)
**Initialize→**
s: source node
dist: an array such that dist(s) = 0, and for all other nodes v, dist (v) = ∞
Q: a queue of all nodes in the graph
S: an empty set
**Run→**
for each node s in nodes:
while Q is not empty:
pop node v from Q that is not in S with the smallest dist(v)
add node v to S
update dist such that for all adjacent nodes u of v (from w(u,v)):
if dist(u) >dist(v) + ee(u):
dist(u) = dist(v) + ee(u)
else:
no change

To enhance GPF and provide tradeoffs between energy savings and network performance, a network dial is designed with the capability to switch between different modes. The goal behind this network dial is to translate the intent of the network administrator to operate the network in a certain energy mode into a distributed systems problem and solving it by fetching the required energy and performance parameters, computing routing metrics as per the dial mode set, running the GPF algorithm, and reconfiguring the network accordingly. The energy savings are determined by one parameter – the current power consumption of the device in Watts, and network performance is determined by three parameters – the available bandwidth in Mbps, the total delay in ms, and the total packet loss in %. Therefore, the network dial should allow tradeoffs between these four parameters – power, bandwidth, delay, and loss. The final weighted formula that combines these four parameters for the purposes of comparing routes is defined in Eq. (1). The final routing metric (ee) uses four constants – c1 through c4, to act as multipliers in the final routing calculation, and a lower metric would be preferred when making forwarding decisions.

$$ee = c1.Power + \frac{c2.10^4}{Bandwidth} + c3.Delay + c4.Loss \quad (1)$$

Here, the energy savings metric is defined as the first term (c1.Power), while the network performance metric is defined as the remaining part of the equation (the addition of the second, third, and fourth terms). The four constants – c1, c2, c3 and c4 – would need to be manipulated over a range such as 0 to 1 in order to implement the different network dial modes.Dividing 104 by the available bandwidth helps differentiate between higher amounts of bandwidth. Otherwise, all available bandwidth values above 1 Mbps would yield a value between 0 and 1, thereby not impacting the final routing metric significantly. The 104 term helps in differentiating between bandwidths smaller than 10000 Mbps (which would yield a value greater than 1) and greater than 10000 Mbps (which would yield a value smaller than 1).Next, to understand the behaviour of how these constants impact the final routing metric ee, statistical analysis was performed to determine for a given pair of values of power, bandwidth, and delay for two devices A and B, which device would have a lower metric when c1, c2, c3 vary from 0 to 1. The loss value and the c4 constant was ignored in this analysis in order to allow visualizing plots in three dimensions.

Also, network performance is considered adequately by including the bandwidth and delay parameters. A sample plot is shown in Fig. 1, wherein black points indicate that the routing metric of device A was lower than B, the yellow points indicate the opposite, and red points indicate that both devices had the same ee value for those particular values of c1, c2 and c3.

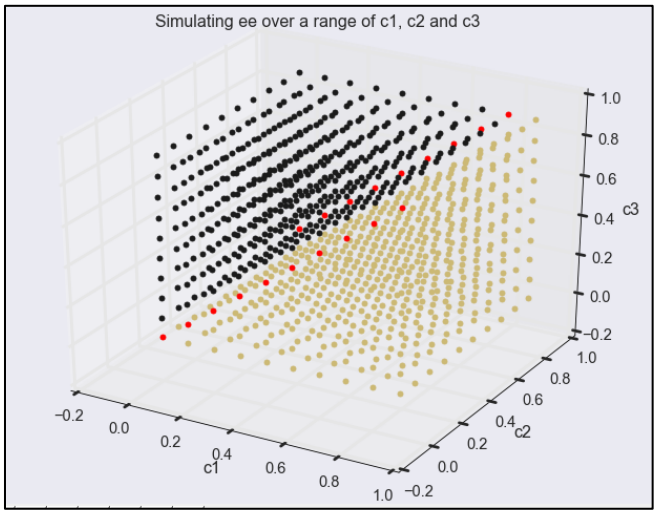


Figure 1. Scatter plot depicting ee values over a range of c1, c2, c3

Simulating over a wide range of power, bandwidth and delay values helped identify a trend of how the constants' values need to be modified in order to get the desired result. Furthermore, the different network dial modes defined are shown in Table I. Since in many instances energy savings come at the cost of network performance, three modes are defined to cover the entire range of operations for the network administrator to select. Although just three dial modes have been defined in this work, the same principles could be used to develop as many modes as required. Implementing these tradeoffs allows for utilizing the opportunity costs associated with energy savings at the expense of network performance, and vice versa.

Table 1. Network dial modes

| **Dial mode** | **Description** |
|---|---|
| 1 | Forwarding decisions are based on energy efficiency only; network performance parameters are not considered. |
| 2 (default) | Forwarding decisions are based on both energy efficiency and network performance parameters. |
| 3 | Forwarding decisions are based on network performance parameters only; energy efficiency is not considered. |

The constants' values chosen for each mode are depicted in Fig. 2. For dial mode 1, c1 is set to 1, and c2, c3, and c4 are set to 0 to ensure that the ee value is calculated by the power consumption term only, and the three network performance parameters (bandwidth, delay, and loss) are not considered. For dial mode 3, c1 is set to 0, and c2, c3, and c4 are set to 1, so that ee is calculated based on network performance only and energy efficiency is not considered. For dial mode 2, c1 is set to 0.75, and c2, c3, and c4 are each set to 0.25. The rationale behind these values is that

since network performance is calculated based on three parameters and energy savings is calculated by just one parameter, the weightage for the four constants should be proportional. Therefore, c1 is set to three times the values of c2, c3 and c4. So, between dial modes 1 and 3, c1 decreases from 1 to 0, and c2, c3, and c4 increase from 0 to 1. These values depicted are one instance of the dial implementation and any values can be set to the constants to create intermediary modes according to specific requirements.

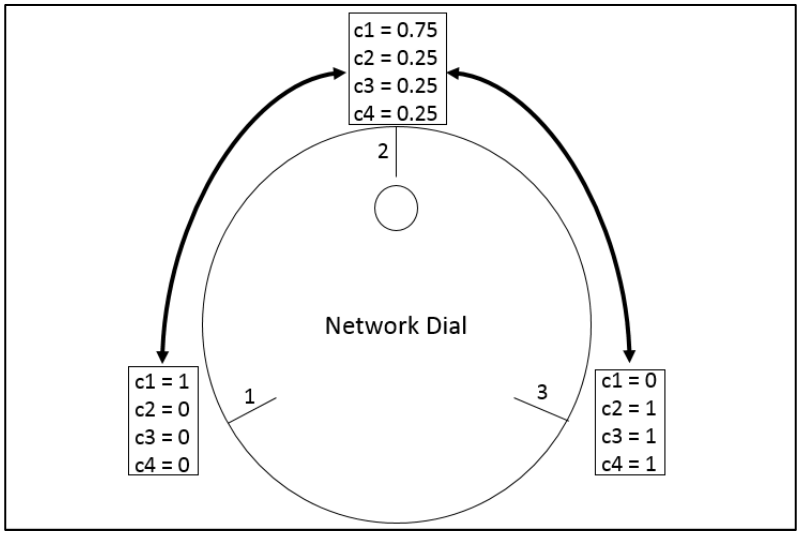


Figure 2. Constants' values in different dial modes for ee metric calculation

The mechanisms to implement GPF provide the implementation details on the technologies employed when working with the different generations of networks. The mechanism details for three types of networking paradigms are provided – traditional, programmable control, and programmable data. These three types represent how networks have evolved over the past decades in an effort to introduce programmability by disaggregating the control and data planes, logically centralizing the control plane, and developing programmable data pipelines.Table 2 provides an overview of the implementation details for each generation.

Table 2. Implementation overview of each networking generation

| **Networking generation** | **Simulated devices** | **Forwarding mechanism** | **Control mechanism** |
|---|---|---|---|
| Traditional | Cisco routers | Static/dynamic routing (OSPF/EIGRP) | SSH |
| Programmable-control | Open vSwitch | OpenFlow match-action pipelines | OpenFlow |
| Programmable-data | Stratum Bmv2 | P4 match-action pipelines | P4Runtime |

The network topology used for the experiments is shown in Fig. 3. It consists of a two-tier Clos architecture with every leaf switch connected to each of the spine switches, and the two leaf switches connected to one host each. Even though the devices used are switches, they are operated as routers with every interface being on a different network in order to allow manipulation of IP traffic. The five spine switches represent the devices used in previous research so that actual power consumption values could be used for the tests [9]. Every switch is connected to an out-of-band network controller which is responsible for making the forwarding

decisions for the traffic flowing between the two hosts. The control connections shown are different for the different types of networks tested.

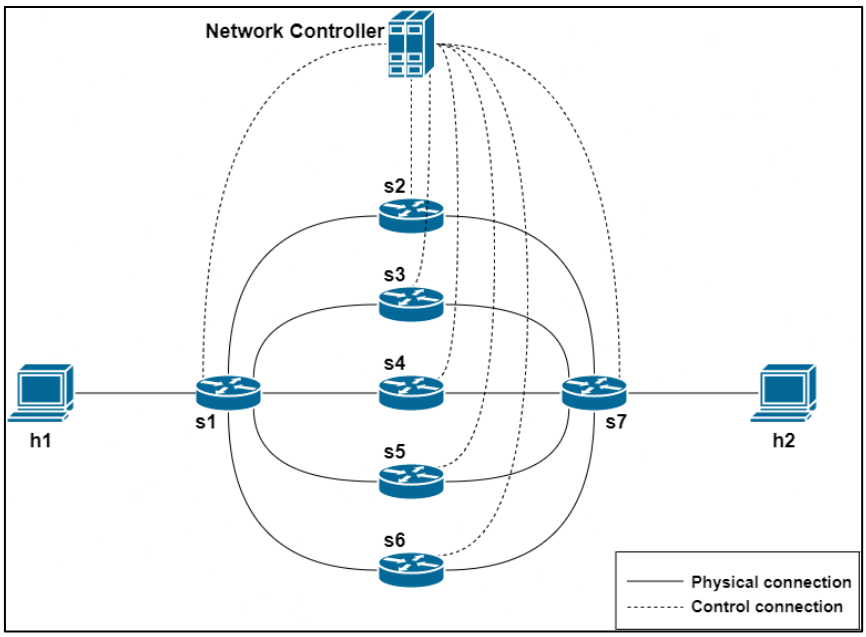


Figure 3. Network topology used in the experiments

The high-level operation of GPF is shown in Fig. 4. The network monitoring tool developed in Chapter III gathers the power, throughput, delay, and packet loss values from the network infrastructure and exposes this information as REST endpoints. The network controller fetches this information using REST endpoints, and also accepts an input on the network dial mode from the user. Next, it runs the GPF algorithm and computes the next hop information for each destination network for each source node. Once the algorithm run is complete, the outputs have to be translated by the control connection module in the specific control connection format – SSH/OpenFlow/P4Runtime. This process is run continuously so that real- time network changes can be transposed onto the network operating behavior. Since the network controller is running on a separate high-CPU server, its processing requirements do not impact the network devices and their ability to process and forward traffic.

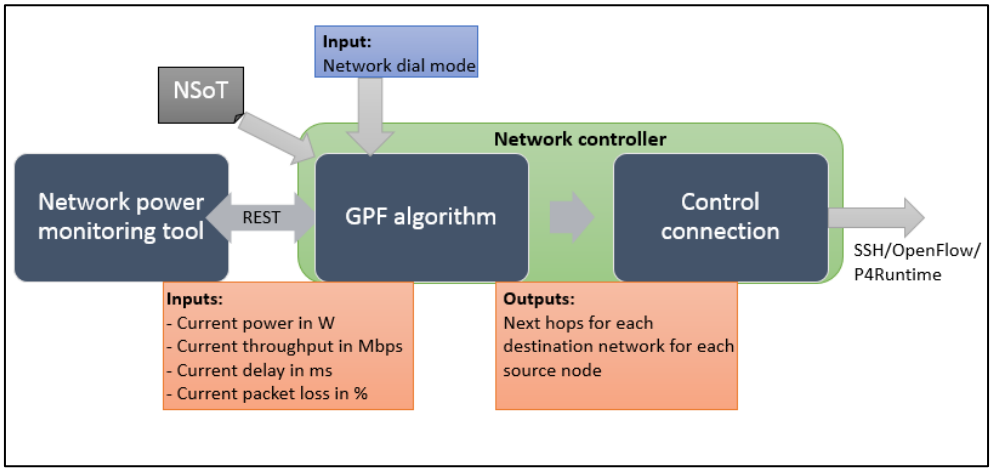


Figure 4. High-level overview of GPF operation

### 3.1. Traditional Network

The traditional network consists of devices with coupled control and data planes, making control decisions locally in a distributed manner, operating protocols supported by the fixed-function data pipelines. The network devices used in the experiment were virtual Cisco 7200 routers simulated in GNS3 [37]. The forwarding mechanisms tested included both static and dynamic routing scenarios; OSPF and EIGRP were employed for dynamic routing testing.

The network controller first reads a NSoT file to discover the devices' information which include their management IP and port. This information is used to send REST calls to the network power monitoring tool to obtain the current power, throughput, delay and loss values. Also, the controller accepts the network dial mode input from the user. Next, the routing metric is computed based on the network dial mode input, and the GPF algorithm is run and its output is a routing table constructed for all source and destination nodes with next hops indicating the path selected based on the dial mode.

SSH was used as the control protocol to configure the appropriate routes onto the devices. The control connection module used a Python library called Netmiko to initiate SSH connections and send routing commands onto the devices [38]. In the case of static routing scenario, the network controller configures static routes on all devices based on the next hops information computed by the GPF algorithm. In the case of dynamic routing with OSPF, the network controller overrides the local routing information computed by each device running Dijkstra's SPF algorithm independently by configuring static routes which have lower administrative distances. In the case of dynamic routing with EIGRP which supports setting a preferred route by influencing its metrics, the network controller uses offset lists to prefer one path over the other.

This process is run continuously with the network controller fetching the current power and network performance parameters every five seconds and reconfiguring the network if required based on the output of the GPF algorithm. A change in the network dial mode also triggers a rerun of this process in order to reconfigure the network based on the new dial mode.

### 3.2. Programmable-control network

Programmable-control networks consist of devices with decoupled control and data planes, running a logically centralized controller, and operating protocols supported by their fixed-function data pipelines. The network devices used in the experiment were Open vSwitch (OvS) devices simulated in GNS3 [39]. The forwarding mechanism tested was the OpenFlow match-action pipelines.

The network controller first reads a NSoT file to discover the devices' information which include their OpenFlow datapath ID (DPID), management IP and port. This information is used to send REST calls to the network power monitoring tool to fetch the current power, throughput, delay and loss values. Also, the controller accepts the network dial mode input from the user. Next, the routing metric is computed based on the network dial mode input, and the GPF algorithm is run and its output is a routing table constructed for all source and destination nodes with next hops indicating the path selected based on the dial mode.

OpenFlow was used as the control protocol to configure the appropriate flow entries onto the devices. The control connection module ran an instance of the RYU SDN controller to enable northbound and southbound interfaces [36]. rest_router application was run on Ryu which exposes a convenient set of API's for setting addresses and routes on the OvS devices. The network controller uses REST API calls to push the appropriate flow entries onto the devices

using OpenFlow Flow_Mod messages matching on the destination network with the action to forward out to its next hop computed from the GPF algorithm.

This process is run continuously on the controller with the network controller fetching the real-time power and network performance parameters every five seconds and reconfiguring the network if required based on the output of the GPF algorithm. A change in the network dial mode also triggers a rerun of this process on the network controller in order to reconfigure the network based on the new dial mode.

### 3.3. Programmable-data network

Until now, the real-time power and network performance parameters were fetched by the network controller from the network power monitoring tool. This information was collected and used to make forwarding decisions in a centralized manner. With the advent of programmable packet parsers that support custom headers or protocols, we tested the exchange of the power and network performance parameters in a distributed manner to test if the devices could make energy-efficient forwarding decisions independently.

The programmable-data network enables the devices to expose programmable parsers along with the match-action tables which allow for the design and development of customized protocols. The network devices used in the experiment were Stratum BMv2 switches [40] simulated using the network emulation software Mininet [41]. Stratum is an open-source next-generation switch operating system that exposes a set of SDN interfaces including P4Runtime [42]. BMv2 is a reference P4 software switch that implements the packet- processing behavior specified by a P4 program [43]. The forwarding mechanism tested was the P4 match-action tables. Two types of approaches were developed – in-band and out-of-band.

The in-band approach involved developing a custom P4 data-plane program with a custom header – Real-time Network Power Protocol (RNPP) – added in between the Ethernet and IPv4 headers in order to transmit the required information between network devices in band. The RNPP header is developed as below –

```
header rnpp {
    bit<16>proto_id;
    bit<16>dst_id;
    bit<16>dpid;
    bit<10>power;
    bit<20>bw;
    bit<10>delay;
    bit<8>loss;
}
```

Forwarding was still based on the IPv4_LPM table while the power and network performance parameters exchanged were stored in a custom table – RNPP_table. The network controller employed P4Runtime in order to – *(i)* fetch the current RNPP values from each device's RNPP_table, and *(ii)* run the GPF algorithm based on the collected values and populate the IPv4_LPM table in order to implement energy-efficient forwarding.

The out-of-band approach involved using P4 Packet_Out and Packet_In messages to allow the exchange of the power and network performance parameters out-of-band. A custom P4 data-plane code was developed that supported adding customized header fields to the Packet_Out and Packet_In messages. The flow of operation is as follows: the network controller, using P4Runtime, first generates and sends Packet_Out messages with the customized header bits

initialized but containing null values from each switch. The switches are configured to broadcast these messages out each port after populating the header fields with their respective power and network performance parameters and stripping the Packet_Out specific bits. Next, switches receiving these custom packets are configured using P4Runtime to forward these packets to the network controller via Packet_In messages after adding the Packet_In specific bits. This allows for the exchange and computation of the energy-efficient information in a distributed manner while ensuring this traffic remains out-of-band from actual data traffic.

## 4. RESULTS AND ANALYSIS

The results from the working of the GPF algorithm and the network dial are presented and analyzed in this section. The experiments were based on power consumption data collected from hardware in the prior work [9]. The first set of tests involved comparing a cost-based SPF with GPF (see Fig. 5). The y-axis of the graph indicates the end-to-end Watt/Mbps value of the selected path by each algorithm. As noted from the figure, since all link costs were set to 1 Gbps in the experiments, the cost-based SPF, by design, randomly selected one of the five paths, while GPF always selected the lowest-power, least-congested path.Watt/Mbps helps in normalizing the energy efficiency of different paths and indicates the energy cost per Mbps of traffic.

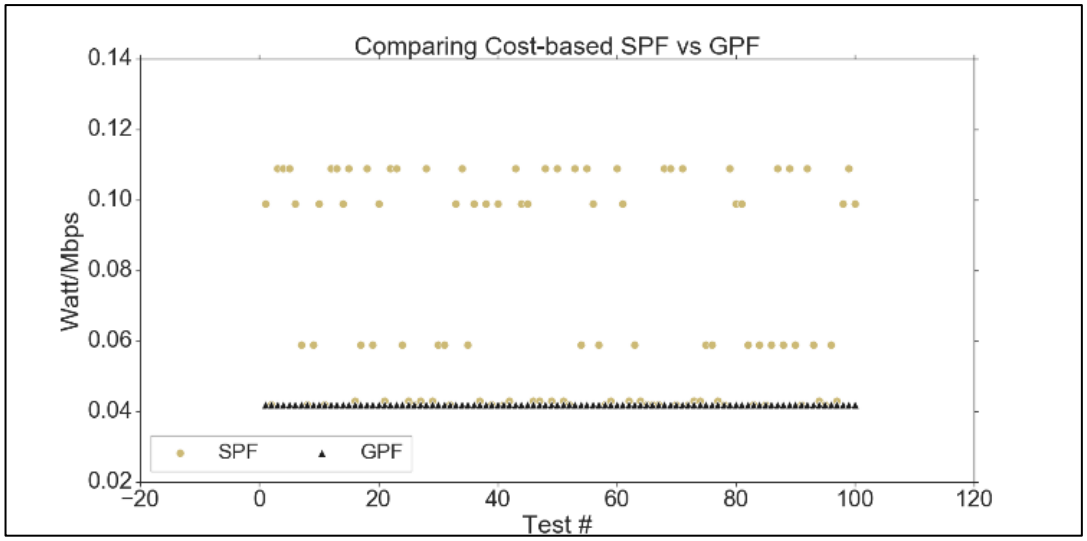


Figure 5. Comparing Watt/Mbps - cost-based SPF vs GPF

Table 3 shows the basic statistical differences between the two algorithms. Both mean and standard deviation values for GPF are lower than that of SPF implying that not only did GPF select low-power paths, but the amount of variation in the power consumption of selected paths was also smaller than that of SPF.

Table 3. Statistical comparison – Watt/Mbps – SPF vs GPF

| Statistic descriptor | SPF – Watt/Mbps | GPF – Watt/Mbps |
|---|---|---|
| **Num. of tests** | 100 | 100 |
| **Mean** | 0.0718 | 0.0398 |
| **Median** | 0.059 | 0.040 |
| **Standard deviation** | 0.0277 | 0.0007 |
| **Min.** | 0.039 | 0.039 |
| **Max.** | 0.109 | 0.041 |

While the differences in the Watt/Mbps values might seem insignificant, when scaled up to the size of production networks handling terabits of traffic, the differences become substantial. Fig. 6 compares the operational energy expenditures in US dollars for 500 devices running cost-based SPF and GPF over different durations. The final dollar values are calculated based on an average electricity cost of $0.137 per kWh in the USA, and do not include other operational costs such as cooling [44].In addition, 1 Watt-hour of network equipment energy savings result in an additional 1.59 Watt-hour of facility-level energy savings [45].

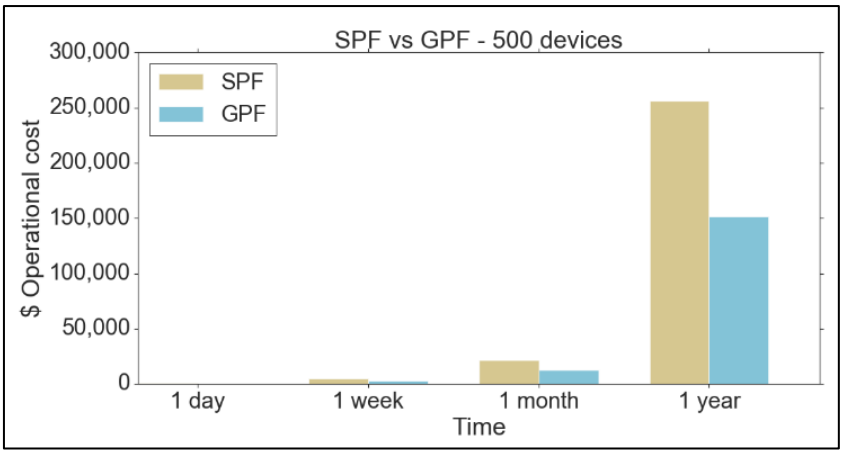


Figure 6. Comparing $ operational cost – SPF vs GPF

To test the network dial implementation, multiple tests were conducted in each of the three modes to check the energy savings and the network performance metrics for the selected path (see Fig. 7). For clarity, a low value for both the energy savings and network performance metrics implies more energy savings and better network performance. For mode 1, low values of the energy savings metric and high values of the network performance metric are noted. For mode 3, high values of the energy savings metric and low values of the network performance metric are noted. For mode 2, intermediate values of the energy savings and the network performance metrics are noted. A broader impact of this dial is that it promotes national security by enabling the government and defense networks to work for longer durations in the event of a disaster, by allowing operations in the high energy savings, low network performance mode.

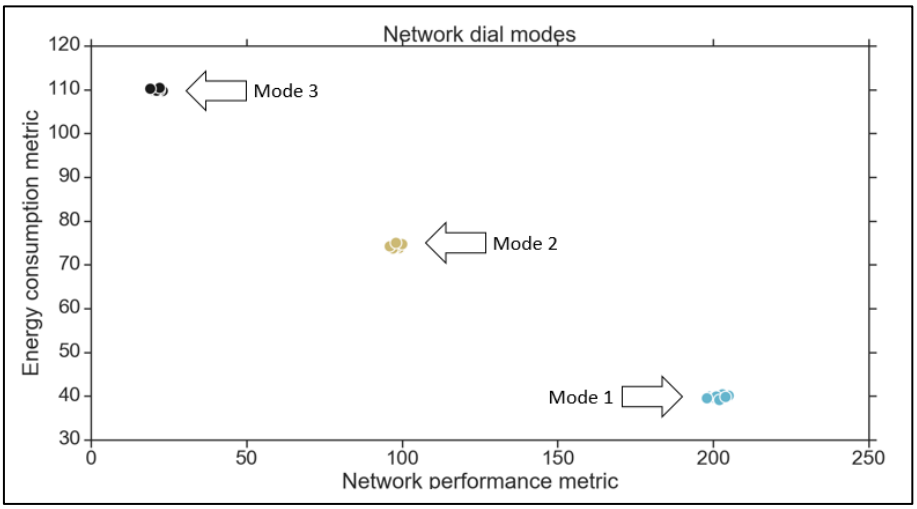


Figure 7. Energy consumption and network performance metrics in different network dial modes

## 5. CONCLUSIONS

A novel technique to incorporate energy efficiency into the network forwarding decisions and to provide tradeoffs between energy savings and network performance was presented in this paper. The GPF algorithm developed from this research computes lowest-power, least-congested paths for all network destinations. The experiment results indicate that significant energy and cost savings could be achieved by prioritizing devices that process traffic more energy efficiently. In addition, a network dial with multiple modes is described that provides a convenient way to tradeoff between energy savings and network performance.The results from the network dial implementation confirm the feasibility of operating the network in different energy modes.The paper also describes implementation mechanisms for different generations of networks – traditional, programmable-control, and programmable-data -in order to demonstrate the potential for industry adoption.

The future directions for this work include employing machine-learning algorithms to enable the dynamic optimization of the dial modes, developing a hardware testbed to implement the algorithm on and test with different network devices, and extending the principles of this research to inter-domain routing.

## REFERENCES


[1] P. Corcoran and A. Andrae, "Emerging trends in electricity consumption for consumer ICT," *National University of Ireland, Galway, Connacht, Ireland, Tech. Rep.,* Jul. 2013.

[2] BP, "Statistical review of world energy," vol. 69, 2020. [online] Available at: https://www.bp.com/content/dam/bp/business-sites/en/global/corporate/pdfs/energy-economics/statistical-review/bp-stats-review-2020-full-report.pdf.

[3] A. S. G. Andrae and T. Edler, "On global electricity usage of communication technology: Trends to 2030," *Challenges*, vol. 6, pp. 117-157, Jun. 2015.

[4] C. Preist and P. Shabajee, "Energy use in the media cloud: Behaviour change, or technofix?" in *Proceedings of the IEEE 2nd International Conference on Cloud Computing Technology and Science*, pp. 581-586, Nov. 2010.

[5] C. Preist, D. Schien, and E. Blevis, "Understanding and mitigating the effects of devices and cloud service design decisions on the environmental footprint of digital infrastructure," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1324-1337, May 2016.

[6] W. V. Heddeghem et al., "Trends in worldwide ICT electricity consumption from 2007 to 2012," *Computer Communications*, vol. 50, pp. 64-76, Sep. 2014.

[7] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," in *Proceedings of IFIP Networking*, pp. 795-808, May 2009.

[8] P. Mahadevan, S. Banerjee, and P. Sharma, "Energy proportionality of an enterprise network," in *Proceedings of the 1st ACM SIGCOMM workshop on Green networking*, pp. 53-60, Aug. 2010.

[9] R. Gandotra and L. Perigo, "Comparing energy efficiencies of SDN hardware based on forwarding configurations," in *Proceedings of the 29th International Conference on Computer Communications and Networks (ICCCN)*, Honolulu, USA, Aug. 2020.

[10] R. Gandotra and L. Perigo, "We've got the power: A framework for real-time network power monitoring," *Journal of Computer and Communications*, vol. 8, pp. 75-88, May 2020.

[11] M. Gupta and S. Singh, "Greening of the Internet," in *Proceedings of ACM SIGCOMM*, pp. 19-26, Aug. 2003.

[12] M. Gupta, S. Grover, and S. Singh, "A feasibility study for power management in LAN switches," in *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP)*, Berlin, Germany, pp. 361-371, Oct. 2004.

[13] L. Chiaraviglio, M. Mellia, and F. Neri, "Energy-aware backbone networks: A case study," in Proceedings of the 1st International Workshop on Green Communications (GreenComm) in conjunction with the IEEE International Conference on Communications, Dresden, Germany, Jun. 2009.

[14] K.-H. Ho and C.-C. Cheung, "Green distributed routing protocol for sleep coordination in wired core networks," in *Proceedings of 6th International Conference on Networked Computing*, pp. 1-6, May 2010.

[15] K. Xie et al., "E$^3$MC: Improving energy efficiency via elastic multi-controller SDN in data center networks," *IEEE Access*, vol. 4, pp. 6780-6791, Oct. 2016.

[16] M. Gupta and S. Singh, "Using low-power modes for energy conservation in Ethernet LANs," in *Proceedings of the 26th Annual IEEE Conference on Computer Communications (INFOCOM 2007)*, Anchorage, Alaska, pp. 2451 – 2455, May 2007.

[17] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate adaptation," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NDSI)*, San Francisco, USA, pp. 323-336, Apr. 2008.

[18] G. Ananthanarayanan and R. H. Katz, "Greening the Switch," in Proceedings of the USENIX Workshop on Power Aware Computing and Systems (HotPower), held at the Symposium on Operating Systems Design and Implementation (OSDI 2008), San Diego, USA, Dec. 2008.

[19] K. Christensen, P. Reviriego, B. Nordman, M. Mostowfi, and J. A. Maestro, "IEEE 802.3az: The road to energy efficient Ethernet," *IEEE Communications. Magazine*, vol. 48, no. 11, pp. 50–56, Nov. 2010.

[20] K. Christensen, B. Nordman, and R. Brown, "Power management in networked devices," *IEEE Computer*, vol. 37, pp. 91–93, Aug. 2004.

[21] C. Gunaratne, K. Christensen, and B. Nordman, "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections and scaling of link speed," *International Journal of Network Management*, vol. 15, pp. 297–310, Sep. 2005.

[22] C. Gunaratne, K. Christensen, and S. W. Suen, "Ethernet Adaptive Link Rate (ALR): Analysis of a buffer threshold policy," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2006)*, San Francisco, USA, pp. 533-534, Nov. 2006.

[23] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "Energy aware network operations," in *Proceedings of the IEEE Global Internet Symposium*, Rio de Janeiro, Brazil, pp. 1-6, Apr. 2009.

[24] M. d. S. Conterato, T. C. Ferreto, F. Rossi, W. d. S. Marques, and P. S. S. d. Souza, "Reducing energy consumption in SDN-based data center networks through flow consolidation strategies," in *Proceedings of 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 1384-1391, Apr. 2019.

[25] K. J. Christensen, C. Gunaratne, B. Nordman, and A. D. George, "The next frontier for communications networks: Power management," *Computer Communications*, vol. 27, pp. 1758–1770, Dec. 2004.

[26] M. Jimeno and K. Christensen, "A Prototype Power Management Proxy for Gnutella Peer-to-Peer File Sharing," in *Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN 2007)*, Oct. 2007.

[27] M. Allman, K. Christensen, B. Nordman, and V. Paxson, "Enabling an energy-efficient future Internet through selectively connected end systems," *Proc. ACM SIGCOMM HotNets Workshop (HotNets 07)*, Atlanta, GA, Nov. 2007.

[28] S. Nedevschi et al., "Skilled in the art of being idle: Reducing energy waste in networked systems," in *Proceedings of the. 6th ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, USA, pp. 381-394, Apr. 2009.

[29] A. Akella, R. Balan and S. Seshan.. "Protocols for low power," *SIGCOMM CCR*, Jan. 2002.

[30] M. Rodríguez-Pérez, S. Herrería-Alonso, M. Fernández-Veiga, and C. López-García, "Improved opportunistic sleeping algorithms for LAN switches," in *Proceedings of the IEEE Globecom*, Honolulu, USA, pp. 1-6, Nov. 2009.

[31] P. Reviriego, J. A. Maestro, J. A. Hernandez, and D. Larrabeiti, "Burst transmission for Energy-Efficient Ethernet," *IEEE Internet Comput.*, vol. 14, no. 4, pp. 50–57, Jul. 2010.

[32] C.-S. Li, Y. Ofek, and M. Yung, "Time-driven priority flow control for real-time heterogeneous Internetworking," *IEEE INFOCOM'96*, Mar. 1996.

[33] M. Baldi and Y. Ofek, "Time for a "Greener" Internet," in Proceedings of the 1st International Workshop on Green Communications (GreenComm) in conjunction with the IEEE International Conference on Communications, Dresden, Germany, Jun. 2009.

[34] B. Heller et al., "ElasticTree: Saving energy in data center networks," in *Proc. of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 249-264, Apr. 2010.

[35] M. Zhang, C. Yi, B. Liu and B. Zhang, "GreenTE: Power-aware traffic engineering," in *Proceedings of the 18th IEEE International Conference on Network Protocols*, pp. 21-30, Oct. 2010.
[36] Github.com. Ryu – Component-based software defined networking framework. [Online] Available: https://github.com/faucetsdn/ryu.
[37] Gns3.com. GNS3 | The software that empowers network professionals. [Online] Available: https://www.gns3.com/.
[38] Github.com. Netmiko – Multi-vendor library to simplify Paramiko SSH connections to network devices. [Online] Available: https://github.com/ktbyers/netmiko.
[39] B. Pfaff et al., "The design and implementation of Open vSwitch," in Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI), pp. 117-130, May 2015.
[40] Github.com. Stratum - Enabling the era of next-generation SDN. [Online] Available: https://github.com/stratum/stratum.
[41] Github.com. Mininet – Rapid prototyping for software defined networks. [Online] Available: https://github.com/mininet/mininet.
[42] P4.org API Working Group, "P4Runtime specification." [Online] Available: https://p4.org/p4runtime/spec/v1.2.0/P4Runtime-Spec.pdf.
[43] P. Bosshart et al., "P4: Programming Protocol-Independent Packet Processors," ACM SIGCOMM Computer Communication Review, vol. 44, no. 3, pp. 87-95, Jul. 2014.
[44] U.S. Bureau of Labor Statistics, "Average energy prices for the United States." [Online] Available: https://www.bls.gov/regions/midwest/data/AverageEnergyPrices_SelectedAreas_ Table.htm.
[45] R. Ascierto and A. Lawrence, "Global data center survey," Uptime Institute, Jul. 2020.