Demystifying the Energy Efficiency of Network Function Virtualization

Zhifeng Xu[1] Fangming Liu*[1] Tao Wang[1] Hong Xu[2]
[1]Key Laboratory of Services Computing Technology and System, Ministry of Education,
School of Computer Science and Technology, Huazhong University of Science and Technology
[2]NetX Lab @ City University of Hong Kong

***Abstract*—Middleboxes are prevalent in today's enterprise and data center networks. Network function virtualization (NFV) is a promising technology to replace dedicated hardware middleboxes with virtualized network functions (VNFs) running on commodity servers. However, no prior study has examined the energy efficiency of different NFV implementations. In this paper, we conduct a measurement study on the power efficiency of software data planes, the virtual I/O and the software middleboxes, which are important parts of the NFV implementations. We run two popular software middleboxes (Snort and Bro) on three common software data planes (i.e., DPDK-OVS, Click Modular Router and Netmap). Our results show significant differences on power among those different NFV implementations. We analyze the underlying design choices and give implications on how to build more power efficient NFV implementations.**

I. Introduction

Middleboxes are ubiquitous in today's data center and enterprise networks for various purposes, such as improving performance (e.g., WAN optimizer, load balancer, etc.), ensuring security (e.g., Intrusion Detection Systems (IDS), firewall, etc.) and providing remote connectivity (e.g., VPNs) [19]. Traditionally middleboxes are implemented as dedicated hardware boxes. They are expensive to build and manage: introducing new network functions means that the operators have to deploy new hardware at the next purchase cycle. Moreover, dedicated hardware can not easily scale up or down according to traffic demands.

Network Function Virtualization (NFV) has emerged as a promising technology to overcome the limitations of traditional middleboxes. NFV implements various network functions in software as virtual machines running on commodity servers, allowing packet processing to exploit the benefits of virtualization. The operator can elastically adjust resource provisioning for her virtual network functions (VNFs), and easily implement new ones. Given such promising benefits, NFV has already gained a considerable momentum, with many prototype development and standardization efforts ongoing [3].

The Corresponding Author is Fangming Liu (fmliu@hust.edu.cn). The research was supported in part by a grant from the National Natural Science Foundation of China (NSFC) under grant No.61520106005, and by a grant from National 973 Basic Research Program under grant No.2014CB347800. Hong Xu's work was supported by grants from the Research Grants Council of the HKSAR, China: GRF 9042179 (CityU 11202315), ECS 9048007 (CityU 21201714), and CRF C7036-15G.

Current research on NFV focuses on performance issues. In this work, we study the energy efficiency of various software data plane technologies for implementing fast packet processing, which has received little attention so far. NFV potentially can consume substantial energy in a large-scale network. It is reported that the number of middleboxes in enterprise networks is comparable to that of L2/L3 forwarding devices [19], and all these network devices account for about $\sim$15% of the total energy consumption of the whole data center [12]. When the hardware middleboxes are replaced with software functions running on commodity servers, their power consumption has not been studied yet.

Since very little is known about the energy consumption of NFV, we set out to explore the following questions in our measurement study. First, what is the energy efficiency of various software data planes, such as Intel DPDK OVS, Netmap, and Click that can be used to develop different VNFs? This is a major source of energy as packet processing is now done by CPU instead of dedicated hardware. Second, what would be the energy overhead of virtualization which is used to host and separate different VNFs? Third, what is the complete energy efficiency profile when we run VNFs such as `Bro` and `Snort` in VMs with software data planes? In answering these questions, our results can be useful in many ways: developers may use them to optimize the energy efficiency of the data plane technologies or the VNFs they write; operators may use them as guidelines to calculate the potential energy saving of NFV in their setting compared to a hardware deployment, choose appropriate software data plane and optimize their VNF deployment or load balancing strategy for energy saving [11, 14, 21].

We conduct a measurement of energy efficiency of three popular software data planes: DPDK-OVS, Netmap VALE and Click, and two software implementations of NFs Bro and Snort on a small-scale NFV cluster. Our main findings are the following:

- Large packets are easy to process and cost less power than small packets. For each data plane, when running at full speed, larger packets cost less power since the whole processing of larger packets is consumed by I/O of most time, CPU usage is relatively low. Batching small packets might be a possible way for data planes to reduce resources usages and power consumption.

- For small packets at low sending rates, power consumption of Click and Netmap VALE are close. However for higher rates, Netmap is more power efficient than Click and DPDK-OVS. For large packets, power consumption of Click and Netmap VALE are close. DPDK-OVS always works with high power consumption even when no traffic to handle. Considering the inefficiency of power, DPDK provides power management APIs to compromise between power consumption and performance.
- Different virtual I/O implementations lead to different power consumptions. Usually, data plane specified virtualizations like vhost-user and ivshmem are more efficient in both power and performance than generic I/O virtualization (e.g., e1000, virtio etc.), because many standardized operations are reduced by data planes. This is a compromise between efficiency and generality.
- For different VNFs, when processing same traffics, Bro costs more power than Snort on DPDK-OVS. On Click, their power consumptions are very close. On Netmap VALE, they consume nearly the same power when processing small packets, while Bro costs more power than Snort when handling large packets.

II. Background

In this section, we briefly review NFV and the-state-of-art software data planes that are developed to achieve high performance of the virtualized network functions.

A. Network Function Virtualization

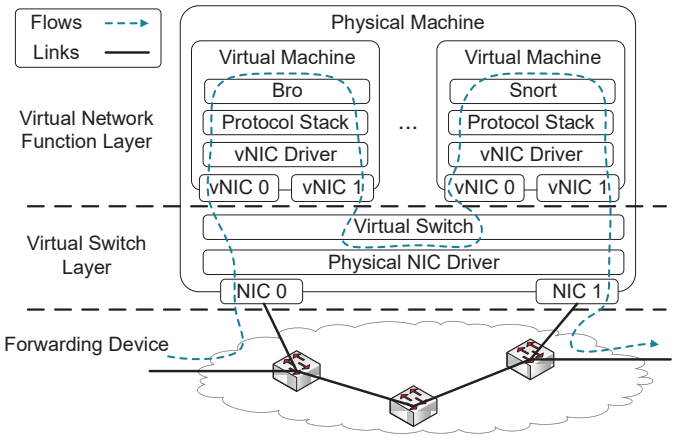

Fig. 1. A NFV deployment where flows are directed through a network to the virtual network functions deployed on the commodity servers.

A typical NFV architecture is depicted in Fig. 1. The packet processing stack can be logically viewed as a three-layer framework [22]: the forwarding device layer, the virtual switch layer and the virtual network function layer. In the forwarding device layer, OpenFlow switches forward flows to physical machines for packet processing according to forwarding rules. The virtual switch layer inside the physical machine is responsible for passing the packets from the forwarding device layer up to the virtual network function layer for processing. The virtual network function layer is composed of the virtual machines that perform specific network functions. Usually, NFV leverages virtualization to accelerate the speed of service deployment and development.

1) System Virtualization: For the reason of high resource usage, flexibility, scalability, performance isolation etc., VNFs run in virtualized environments. Hypervisors like kvm, Xen, VMWare ESX or Microsoft Hyper-V can provide required environment to run NFV applications, but they inherit the overhead of hypervisors [23]. Containers like Docker and LXC provide light-weighted runtime environments for NFV, but they provide weak isolation and introduce high complexity of management.

2) Network I/O Virtualization: To let the virtual NICs (vNICs) of VMs to get/put data from/to the physical NICs (pNICs) and communicate with other vNICs, hypervisors usually support I/O virtualization. When packets first arrive at pNICs of the host, hypervisors get those traffic data from the host network stack and then handed them to the right VM. This procedure causes high overhead and leads to low performance. Technologies like direct memory access (DMA), PCI passthrough and Single root I/O Virtualization (SR-IOV) can accelerate this host to guest path, but need hardware support. Software technologies like huge page memory mapping, batch processing, zero copy, etc. also help to accelerate the packets delivery from host to guest.

B. Software Data Plane

TABLE I
Summary of Data Plane

Data Plane	Description
DPDK-OVS [2]	A DPDK accelerated version of Open vSwitch.
Netmap [18]	A high speed packet I/O framework that applications can run on top of it.
Click [13, 15]	A well-known modular software router.

Except for the virtualization, another important technology NFV exploits is software packet processing, which we introduce here. The TCP/IP stack in the Linux kernel has poor throughput and cannot satisfy the NFV's demand of performing complex packet processing at line rate [8]. This problem has attracted much attention recently, and several software data planes (i.e., virtual switch layer) have been proposed to deliver near bare-metal packet processing performance [2, 4, 18]. In our measurement study, we choose three popular open source software data planes as listed in the Table I.

DPDK-OVS. DPDK-OVS [2] is the Intel DPDK accelerated Open vSwitch. It takes advantages of DPDK [1], which provides a set of libraries and drivers for fast packet processing. With the user-space drivers provided by DPDK, applications (e.g., L3 forwarding, IPsec, firewall, etc.) can put/get data directly to/from DMA area while in-memory data copies are

avoided. And DPDK Poll Mode Drivers (PMD) are used to eliminate overhead of context switching caused by interrupts in linux kernel. Also, DPDK provides faster inter-process communication facility built upon shared memory and user-space locking. By using hugepages, minor page faults can be avoided. DPDK vSwitch also moves the software switch from kernel space to user space, facilitating industry and proprietary enhancements.

Netmap. Netmap [18] is a framework for high speed packet I/O. It uses some well-known performance-boosting techniques, such as memory-mapping which maps the NIC's packet buffers to the main memory, I/O batching and circular buffer queue. Netmap implements the APIs of *libpcap*, so many applications can run on top of it without modification. Netmap can reach line rate on 10G NICs (14.88 Mpps). Other frameworks (e.g., DPDK, DNA) achieve similar speeds but lack the ease of use and portability.

Click. Click is a well-known modular software router [13, 15]. Click consists of many predefined elements written in C++, each of which can perform certain actions on a packet. By connecting these elements in a certain manner, developers can build packet processing pipelines to implement their network functions.

III. Measurement Methodology

In this section, we describe our measurement methodology to analyze the energy efficiency of software data planes and the VNFs. We first present the testbed on which our measurements are conducted. Then, we introduce the three software data planes, two kinds of I/O virtualization technologies and two types of VNFs in our measurements. Finally, we run some tests to measure the performance of data planes under different settings in order to set proper parameters for our power measurements.

A. Measurement Platform

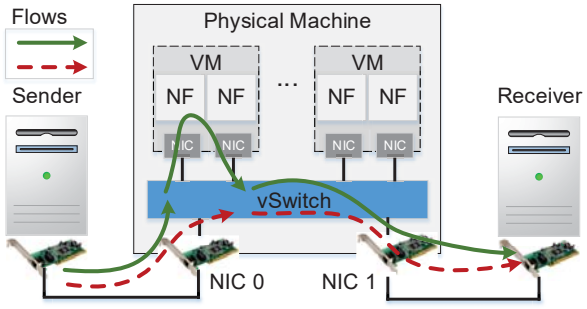

Fig. 2. The measurement platform with three servers of which the left one is acting as traffic sender, the right one is traffic receiver and the middle one is deployed with software data plane and the VNFs run on top of it.

We perform all the measurements on three servers, each has two 8-core Intel Xeon E5620 2.4GHz processors, 16GB memory, and two Intel 82574L Gigabit NICs. All servers run 64-bit CentOS 7. As shown in Fig. 2, in our testbed, two servers act as sender and receiver, one runs the software data plane and VNF applications. All servers are connected to an OpenFlow switch, which routes all packets from the sender to the receiver through the NFV server.

On the sender server, we use `tcpreplay` [6] to send traffic. We choose `tcpreplay` since it can read packets from a `pcap` file generated from real-world traces and send them to the network through a specified interface. With the '`-p`' or '`-m`' option, we can control the sending rate of these packets. Thus we can ensure that all data planes and VNFs on the NFV server process identical amount of traffic. At the receiver side, we use `tcpdump` to capture all packets received by the NIC and compare them against the source packet trace to validate whether the VNFs work correctly.

A power meter is attached to the NFV server to monitor its power consumption at the granularity of every second. We send packets at different rates, lasting for 30 seconds in each set of experiments. A sample output of the power meter readings is shown in Fig. 3. We can see that when processing packets at a fixed rate, the power consumption of the NFV server increases and then stays stable at certain value (fluctuations within 2W). When processing finished, the power consumption drops to normal value. Excluding the increasing and decreasing stages, we take the average value of the stage when power consumption stays stable as the results.

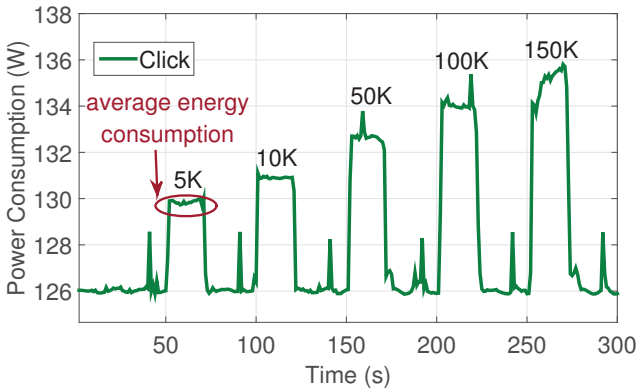

Fig. 3. Sample output of power meter for Click when handling different packet arrival rates with packets of different bytes (in kpps).

B. NFV setup

As we have introduced in Sec. II, when packets arrived at the pNIC of the NFV server, they are redirected to the VM that hosts the specific VNFs. The processing stack includes three layers: 1) the virtual switch layer, 2) the virtual I/O layer and 3) the VNF layer. Thus, to fully understand the energy characteristics of all three layers, we design our measurement study that focuses on the following scenarios: 1) raw data plane, 2) data plane with virtual I/O, and 3) data plane with VM running VNFs.

If not stated otherwise, through all our measurements, we build DPDK-OVS, Click, and netmap VALE on the NFV server separately in each run. VMs are allocated with 2 logical

CPU cores and 4GB memory. The `irqbalance` service on the NFV server is turned off to prevent the load from being distributed to multiple cores, and the Linux `taskset` tool is used to enforce VNFs to use the same number of CPU cores. All the measurements are carried out independently for 10 runs, and we show the average values as results.

In the following we explain the detailed setup for each of the three scenarios we investigate.

*1) **Raw Data Plane**:* In this scenario, we consider a simple measurement setting. As depicted in the Fig. 2, once a packet arrived at a pNIC (i.e., NIC 0) of the NFV server, the data plane (i.e., virtual switch) will send it out through the other pNIC (i.e., NIC 1). As shown by the dashed line in Fig. 2, packets only traverse through the data plane without virtual I/O or VNF processing.

We present the detailed setting of each data plane in this scenario. For DPDK-OVS, the two physical NICs of the NFV server are binded to DPDK driver, and the *vswitchd* process is compiled with DPDK support. For Click, we use the `FromDevice` (i.e., NIC 0), `ToDevice` (i.e., NIC 1) and `Queue` elements to implement the data plane. For Netmap, we attach the two physical NICs to the VALE switch. The VALE switch is written with Netmap APIs and works as a learning bridge, so no extra configuration is needed.

*2) **Data plane with virtual I/O**:* In order to understand how the virtual I/O layer in NFV affects the energy efficiency, the second set of measurements is carried out for comparison with the first one. In this scenario, VMs are attached to the data plane (i.e., virtual switch) through virtualized network interfaces. Once packets arrived at a physical NIC, they are redirected to the VM through a I/O virtualization layer. The packet datapath is depicted by the solid line in Fig. 2. It is worth noting that, in this setting, the VM does not provide any network functions for packet processing.

Through the comparison with raw data plane results, we can infer the energy efficiency of I/O virtualization layer. We consider two kinds of I/O virtualization: generic virtualization and data plane specific virtualization. If using generic virtualization technologies (e.g., e1000, rtl8139, virtio, etc.), VMs can communicate with all data planes. A packet data is sent from the host kernel to the user space hypervisor process, then to guest process in its kernel space; there are two context switchings and data is copied twice, and vice versa. Among the generic virtualization technologies, virtio has better performance than others and it is now the main platform for I/O virtualization for hypervisors. A guest running virtio-net driver shares a number of `virqueues` with the QEMU process, thus the context switch and data copy between QEMU and the guest can be saved. We choose `virtio` in the measurement as the representative generic I/O virtualization technology.

For data plane specific virtualization, as far as we know, only DPDK provides `vhost-user` and `IVSHMEM` that can be used for host-guest communication. When using these virtual I/O technologies, the VM can only communicate with DPDK-OVS data plane, i.e. DPDK-OVS. `Vhost` offloads the servicing of `virtio-net` to kernel module, reducing context switching and data copies in the virtual dataplane. `Vhost` can support DPDK based applications as well as legacy `virtio` based applications. The `IVSHMEM` library facilitates fast zero-copy data sharing among the host and guest by mapping the memory of the guest directly to the host memory. We choose virtio-net and vhost-user from the two kinds to compare the impacts of them on power.

*3) **Data plane with VNFs**:* The experimental setting is the same as the second scenario except that the VM hosts several different VNFs such as firewall, intrusion detection systems, etc. In this scenario, we run `Bro` [7] and `Snort` [5] in the VM separately. Those two NFs are two widely used software intrusion detection systems [10]. By comparing the power consumption of the VM with/without VNFs running, we can infer the affect of VNFs on power. We can also learn the power consumption of different VNFs.

C. Performance of Different Data Planes

Before we report the results of our measurement study on energy efficiency, we conduct simple experiments to test the basic packet forwarding performance of each data plane. The results are used to set parameters in Sec. IV.

We first run an `iperf` test between the sender and receiver to test the throughput of raw data planes. We use `tc` to control the sending rate at 100Mbps, 200Mbps, etc, and use `iperf` to measure the TCP bandwidth and RTT jitter (with UDP). The results are shown in Fig. 4 and Fig. 5. We can see that all data planes can achieve line rate and the network condition is stable according to the jitter. We conclude that the NFV server with raw data plane is transparent to the sender and receiver.

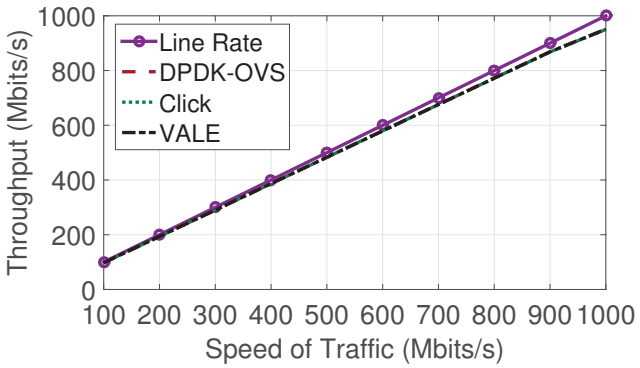

Fig. 4. Bandwidth between the sender and receiver.

Fig. 6 shows the maximum processing capabilities of different raw data planes with different packet sizes. We can see that DPDK-OVS can reach the line rate for all packet sizes. When handling large packets with size 512B, 1024B and 1500B, all data planes can achieve line rate. For small packets like 64B packets, performance of Click and Netmap VALE is poor though it is reported that netmap can achieve line rate even with small packets [18]. We find that in our experiments, the kernel always crashes when using netmap to handle more than 800 kpps 64B packets. We have reported this

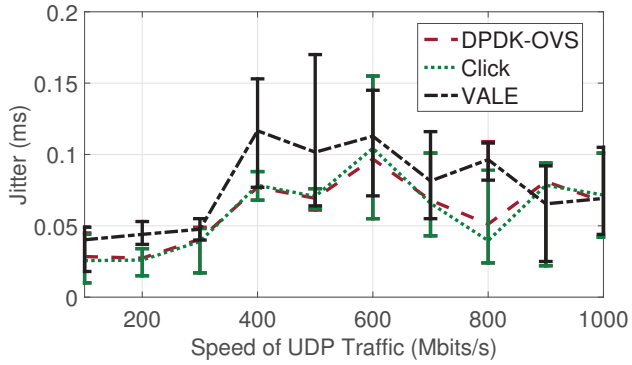

Fig. 5. UDP jitter between the sender and receiver.

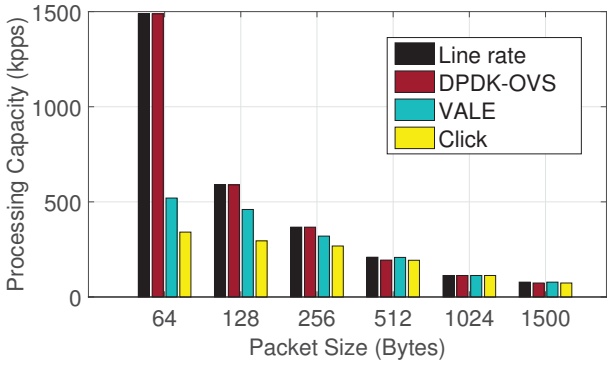

Fig. 6. Performance of different raw data planes.

to the authors of Netmap. Thus we have to limit the the rate of 64B packets below 800 kpps in the following experiments.

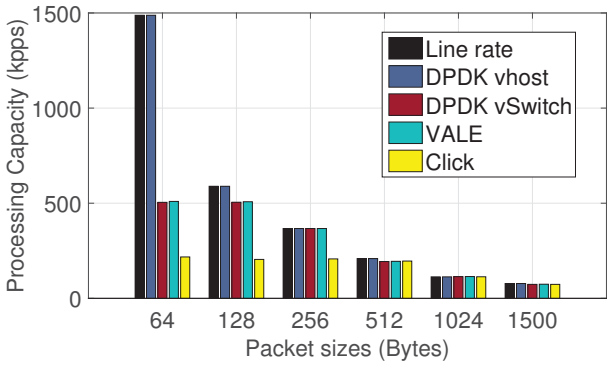

Fig. 7. Performance of different data planes with virtual I/O.

With VMs at the NFV server, packets flow in/out of the VM through a virtual I/O layer. We measured the performance of different data planes with virtual I/O. As we mentioned in Sec. III-B2, we use `virtio-net` and `vhost-user`. The results are shown in Fig. 7. We can see that DPDK-OVS with `vhost-user` can achieve line rate as in the raw data plane case. However with `virtio-net`, DPDK-OVS can only achieve about 500 kpps, because virtio becomes the bottleneck due to its context switching and data copies. Compared with Fig. 6, we find that performance of Click drops from 368 kpps to 217.6 kpps for 64B packets with virtual I/O. So does 128B packets and 256B packets. For 512B, 1024B, and 1500B packets, virtual I/O does not affect the performance.

IV. Evaluation

In this section, we evaluate the power consumption of different NFV implementations, including different data planes, different virtual I/O technologies, and different VNFs. We conduct measurement experiments in three scenarios introduced in Sec. III-B. In each scenario, we plot the power consumption of the same data plane under different traffic loads (i.e., different packet sizes and different packet sending rates), and the power efficiency of different data planes when handling the same load.

A. Data Plane Power Consumption

We first look at power consumption of data plane, without virtual I/O or VNFs. At the sender, we vary the packet size from 64B to 1500B, and for each packet size we vary the sending rate up to the maximum rate for achieving 1Gbps throughput on the NIC. For example, for 256 bytes packets, we choose sending rates of 10 kpps, 50 kpps, 100 kpps, 200 kpps, 300 kpps, 400 kpps, because its maximum sending rate is 367 kpps. While for 1024 bytes packets, the sending rates are 5 kpps, 10 kpps, 50 kpps, 80 kpps, 100 kpps and 120 kpps. The packet arriving rate on pNICs of the NFV server is equal to the sending rate of the sender because

1) Single data plane analysis: We first compare the power consumption of the same data plane under different traffic loads.

For DPDK-OVS, due to the feature of DPDK's Polling Mode Driver (PMD), once the first DPDK port is added to vswitchd process, it creates a polling thread and polls DPDK device in continuous loop. Therefore CPU utilization for that thread is always 100%, and the power consumption raises to about 138 Watt, regardless of the traffic loads, as shown in Fig. 10. For Click and VALE, different traffic incurs different power.

Varying sending rates. The results for Click and Netmap VALE are shown in Fig. 8 and Fig. 9, respectively. We can see from the figures that when handling packets in the same size, the power consumption increases with sending rate. When sending rate exceeds a certain value, the power consumption stays at the peak value. This is because that the sending rate exceeds the maximum processing speed of the data planes with the allocated resources.

For Click, we can see that when sending rates of 64B packets increase from 10 kpps to 600 kpps, the power consumption increase from 127.6W to 139.5W. The 12W increase is same with the power increase of running a core from 0% to 100%. While for 512B packets, 1024B packets and 1500B packets, the increase of power is only 6 to 7W, which means that the CPU core is not fully utilized, and the bottleneck is in the I/O.

For Netmap VALE, the power consumption does not increase with the sending rates in some cases, because Netmap loads a kernel module, its power consumption may be affected by other modules. When handling 1500B packets, the power consumption is the lowest, both in Click and VALE. Because for large packets, most time is consumed by I/O (passing data from the NIC to the kernel), and CPU is relatively idle.

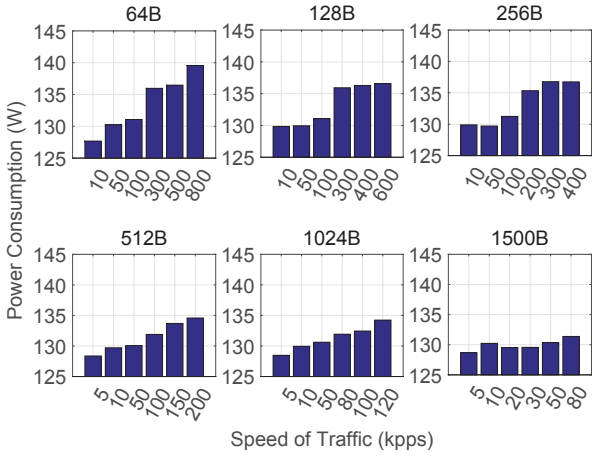

Fig. 8. Power consumption of Click with different loads.

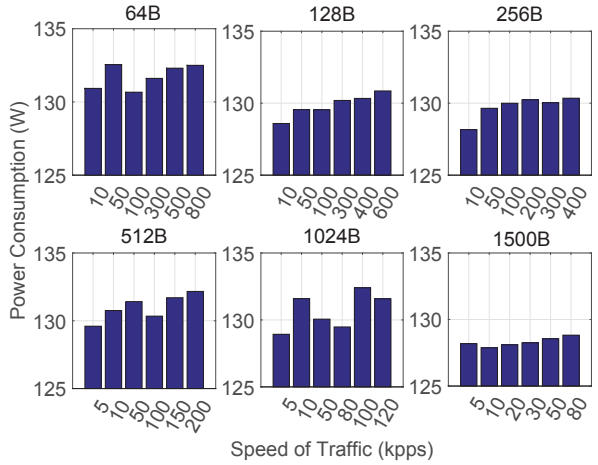

Fig. 9. Power consumption of Netmap with different loads.

Different packet sizes. To see the impact of packet sizes on the power consumption, we send packets with sizes from 64B up to 1500B, at the speed of 50 kpps. We choose 50 kpps as the sending rate since the maximum sending rate of 1500B packets is limited at 79 kpps by the 1GbE NIC. The results are shown in Fig. 11. We can see that as the packet size increases, the power consumption does not increase, indicating that packet size has little impact on the power consumption. The reason is that packet forwarding only involves header parsing and checking, which has nothing to do with packet size.

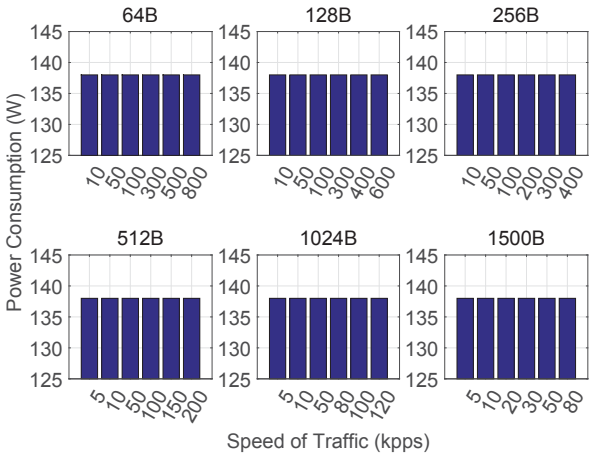

Fig. 10. Power consumption of DPDK-OVS with different loads.

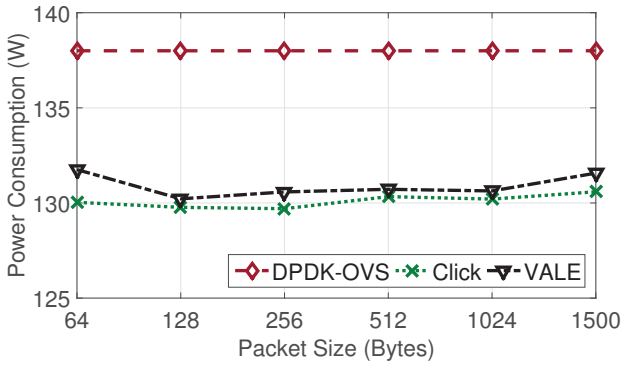

Fig. 11. Power consumption of raw data planes when handling different packet sizes at 50 kpps.

Same amount of bytes in different packet sizes. Even when a data plane processes the same amount of bytes per second, the power consumption may be different when packet sizes are different. For example, sending packets with size 64B at the rate of 100 kpps and sending packets with size 128B at the rate of 50 kpps, the amount of bytes processed per second is the same, while the power consumption may be different. Fig. 12 and Fig. 13 show the differences in power consumption when handling the same amount of bytes. We can see that smaller packets always cost more power than larger packets. By observing the resource utilization, we notice that for Click, when handling 64B packets at 100 kpps, the CPU load is 54.8%, whereas when handling 128B packets at 50 kpps, the CPU load is 41.5%. When processing 64B packets at 200 kpps and 128B packets at the rate of 100 kpps, the CPU loads are 84.7% and 55.1%, respectively. Smaller packets require more CPU because when the amount of bytes is the same, smaller size means larger packet amount, which leads to higher CPU utilization.

2) Comparison between data planes: We then compare the power consumption of different data planes when handling

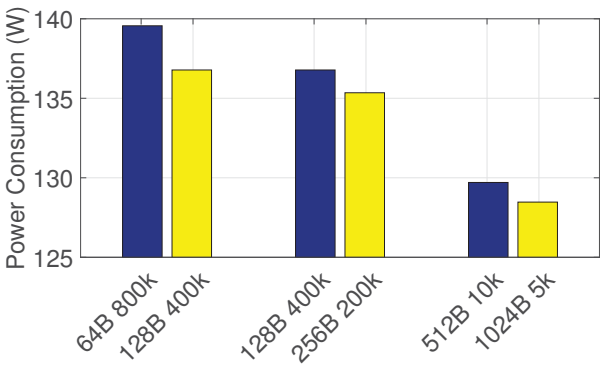

Fig. 12. Power consumption of Click when handling the same amount of bytes in different packet sizes.

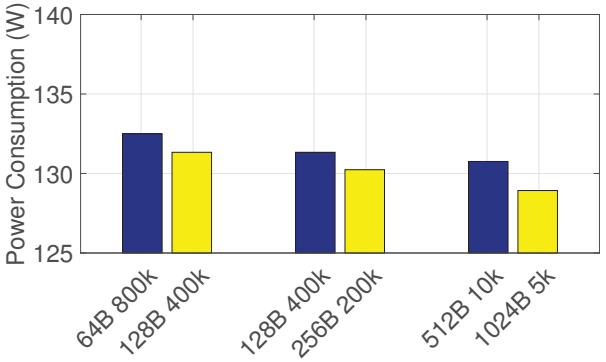

Fig. 13. Power consumption of Netmap VALE when handling same amount of bytes in different packet sizes.

the same traffic loads. The result is shown in Fig. 14. From the first three sub-figures, we can see that for 64B, 128B and 256B packets, when sending rates were relatively low (the first 3 rates), the power consumptions of Click and VALE are close. For higher rates (the last 3 rates), power consumptions of Click continue to increase while VALE barely increases. We can conclude that when handling small packets at high rates, VALE is more power efficient than Click and DPDK-OVS. When packets are sending at low rates, power consumptions of Click and VALE are close. For 512B, 1024B and 1500B packets, the power consumptions of Click and VALE are about 4W to 7W lower than that of DPDK-OVS, because for large packets, the arriving rates are usually slow, while DPDK-OVS keeps polling at high frequency, which is unnecessary and cost more power.

B. Impact of virtual I/O on power

To understand the impact of virtual I/O on power, we investigate the power consumption of VM with virtual I/O. Based on the raw data plane settings above, we start a VM and attach it to the data plane through different virtual I/O. We just need the packets to pass through the virtualization layer and reach to the VM. The VM does not run any applications. Like what we did in the raw data plane scenario, the sender

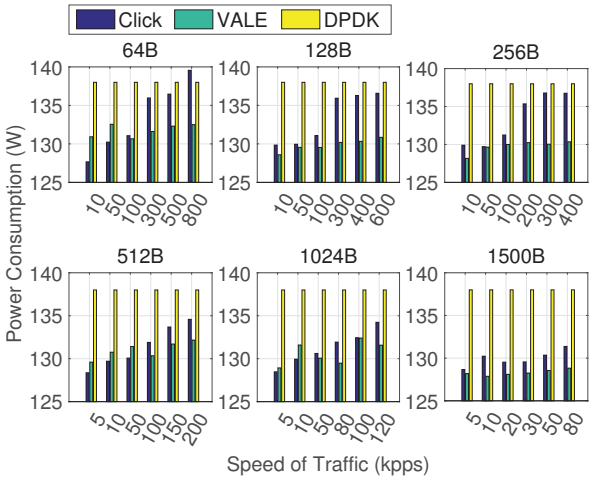

Fig. 14. Power consumption of different data planes when handling same size packets at different rates.

sends packets in different sizes at different rates to the NFV server.

Fig. 15 shows the power consumption of data planes with VM. The result is similar to that of the raw data plane. For DPDK-OVS and Click, power consumption increases with sending rates. For VALE, the power consumption does not increase with the sending rates in some cases.

For 64B, 128B and 256B packets, when sending rates are relatively low, Click costs less power than DPDK-OVS. When sending at higher rates, Click and DPDK-OVS use roughly same power. For 512B, 1024B and 1500B packets, power consumption of DPDK-OVS is always higher than that of the Click. VALE costs less power than Click and DPDK-OVS, and the gap becomes larger when processing larger packets.

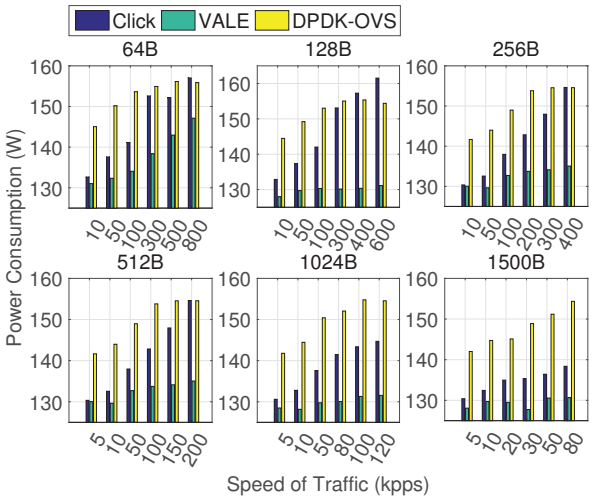

Fig. 15. Power consumption of different data planes with virtual I/O.

Fig. 16 shows the power consumption of virtual I/O when processing different sizes of packet at 50 kpps on different data planes. The power consumptions show little fluctuation

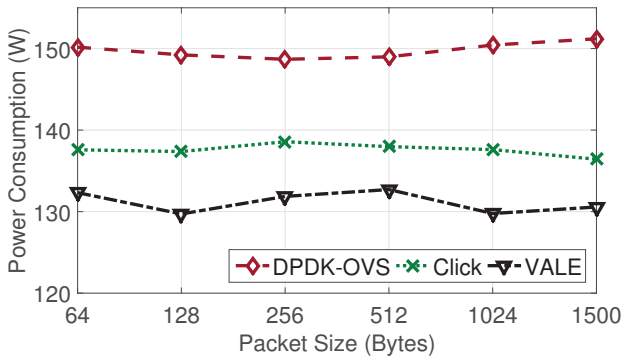

Fig. 16. Power consumption of data plane with virtual I/O. We vary packet sizes at 50kpps.

as the packet sizes increase. We can infer that the packet sizes do not affect the power consumption of the data plane and virtual I/O. In Fig. 11, the power consumptions of Click and VALE are very close. While in Fig. 16, after they both add a virtual I/O layer, they have about 5W power consumption gap. This indicates that virtual I/O causes less power increase on VALE than that on Click, when processing packets at 50 kpps.

By comparing the power consumption of the NFV server before and after virtual I/O is added, we can learn the impact of virtual I/O on power. Fig. 17 shows the gap between the two scenarios. This gap is caused by the VM and the virtual I/O. According to our observation, starting an idle VM does not increase the power consumption of the NFV server. So we consider that the gap as the power consumption of virtual I/O.

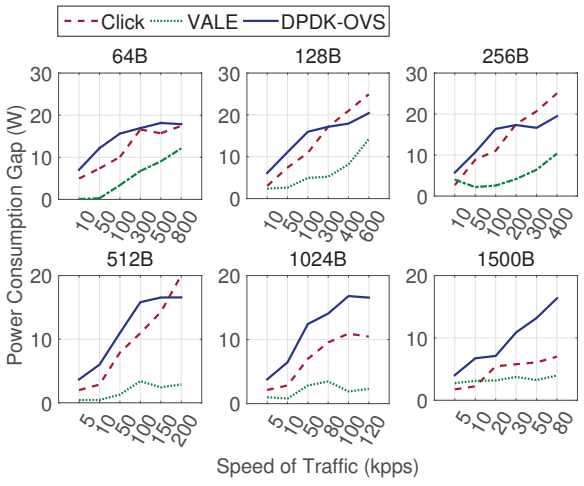

Fig. 17. Power consumption gap caused by virtualization on different data planes when handling different loads.

From Fig. 17, we can see that virtual I/O causes larger power increase on DPDK-OVS and Click than that on VALE. VALE has better support for virtual I/O than DPDK-OVS and Click. Because when sending packets into VM through virtual I/O, Click and DPDK-OVS need to pass data to QEMU first, then the VM get the data from QEMU. However VALE maps the memory of the guest to the host, guest can get the packets by reading specific memory areas, and no data copy is needed.

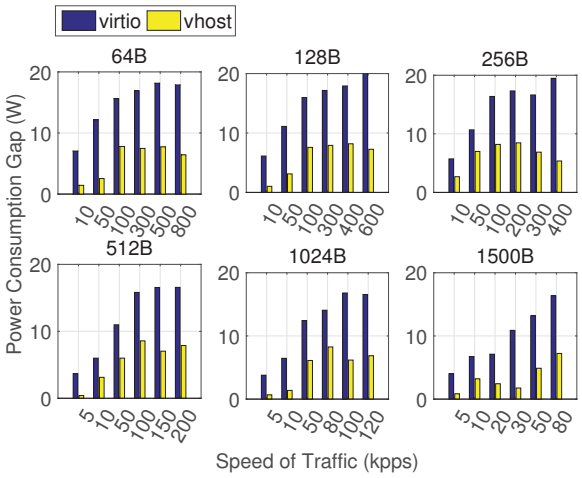

Fig. 18. Power consumption caused by virtio and vhost with different loads.

Fig. 18 shows that power consumption caused by virtio-net increases with the sending rates. Virtio-net needs to copy data from host kernel to the user space QEMU process. Higher rates means more data copies and more power. Vhost caused less power increase than virtio-net, because vhost offloads the servicing of virtio-net to kernel module, which leads to less context switch than virtio.

C. Impact of VNFs

Based on the VM setting above, we run some middleboxes in the VM to process flow in packets. We run two popular software middleboxes Bro and Snort separately and analyze their power consumption characteristics.

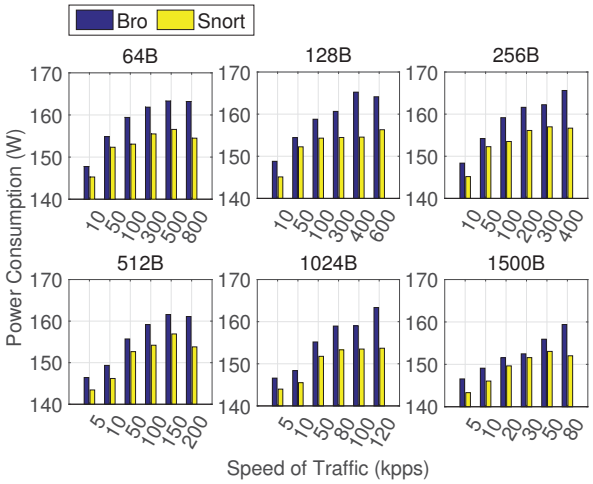

Fig. 19. Power consumption of Bro/Snort running on DPDK-OVS when handling different loads.

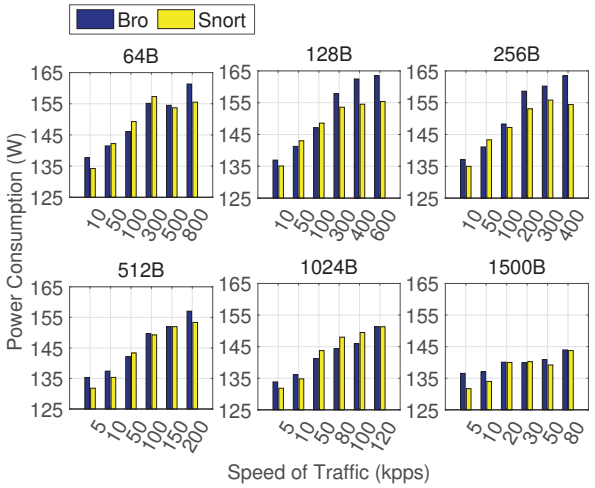

Fig. 20. Power consumption of Bro/Snort running on Click when handling different loads.

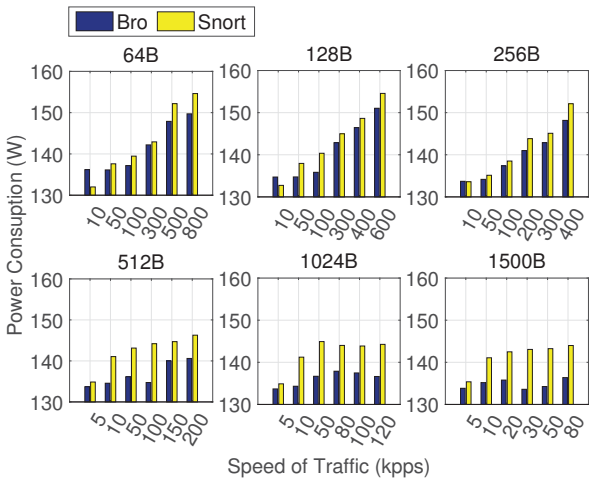

Fig. 21. Power consumption of Bro/Snort running on Netmap when handling different loads.

Fig. 19, 20 and 21 show the power consumption when running Bro and Snort on different data planes. For VALE, when packet sizes are 64B, 128B and 256B, power consumptions of Snort and Bro are very close. While for larger packets, Bro costs more power. For Click, power consumption of Bro and Snort are roughly the same except for several cases. For DPDK-OVS, Bro always costs more power than Snort. When running Bro in the VM with no traffic flow in, the CPU utilization is 0.7%, while for Snort, it is 27% because Snort needs to dump packets from the NIC of the guest all the time.

Snort works at packet logger mode and only checks the packet header for protocol analysis. Bro works at packet dump mode, it will save the packets and analyze the packet headers. Fig. 22 and Fig. 23 show the power consumption of Snort/Bro processing different packets sizes at 50 kpps on different data planes. We can see that power consumptions of Snort and Bro are roughly the same on DPDK-OVS and Click, and it has

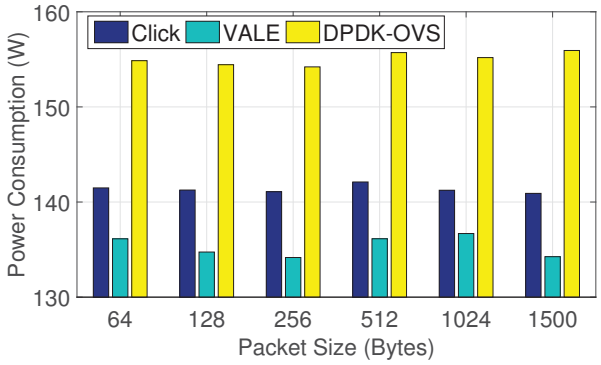

Fig. 22. Power consumption of Bro processing different sized packets at 50 kpps on different data planes.

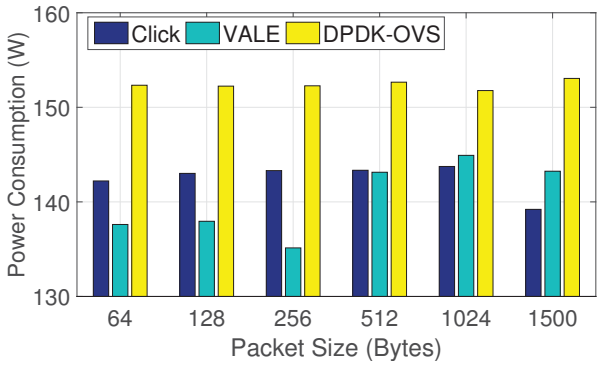

Fig. 23. Power consumption of Snort processing different sized packets at 50 kpps on different data planes.

little fluctuation with the increase of packet sizes.

V. Related work

There are many papers about data center energy, like [24, 25], but works about the NFV energy is very rare. In this section, we briefly survey related work on NFV measurement, as well as on NFV energy management and scheduling to which our findings may be useful.

NFV measurement. Rahul et al. [17] report that the number of middleboxes is comparable with the forwarding devices in enterprise networks. Meanwhile, the failures of middleboxes are common. One of the most important reasons is overload, which is consistent with our findings in Sec. III-C. Wu et al. [22] present PerfSight, a system that can diagnose the degradation of software data planes, which mainly focus on throughput. Despite the preliminary results on performance presented in Sec. III-C, our work mainly investigates the energy efficiency which has drawn much attention in today's industry and literature.

Energy management. Rashid et al. [16] use Bell Labs G-WATT tool to estimate the energy savings that NFV expected according to its models. They focus on three specific use cases and show the possibilities of using NFV to save energy. Different from estimating energy consumption by power

models, Tang et al. [20] present a way to disaggregate energy consumption from the aggregate energy, the results of the disaggregation are used to guide the design of power saving strategies. Dong et al. [9] use Shapley Value to estimate energy consumptions of each mobile applications in a phone to schedule those mobile applications energy efficiently. These methods can also be used in NFV to estimate the power consumption and give guidance to efficiently deploy NFVs. Different from their works that apply power model or disaggregate power consumption, we investigates the details of each part in NFV and look into the power efficiency of those 3 vital components.

Multi-resource scheduling. Multi-resource scheduling in NFV has been a hot topic in today's literature [11, 14, 21]. As energy has become a main concern [20], it should be considered as another dimension in [11, 14, 21]. Our findings present preliminary results on how to choose appropriate underlying software data planes when considering energy consumption, and further be used to design energy-efficient multi-resource scheduling.

VI. Conclusion

NFV replaces traditional hardware middleboxes with software virtual network functions running on commodity servers. This potentially can consume substantial energy in large-scale networks. Current research on NFV focuses on performance, and little attention is paid to the energy efficiency of NFV. In this work, we take the first step to investigate the power efficiency of different NFV implementations. We run two popular software middleboxes (i.e., Bro and Snort) on three common software data planes (i.e., DPDK-OvS, Click and Netmap VALE). Our results show the power consumption differences of different data planes, different I/O virtualization and different VFs. We analyze the underlying reasons and give implications on how to build more power efficient NFVs.

References

[1] Intel Data Plane Development Kit. http://www.dpdk.org/.

[2] Intel DPDK vSwitch. http://www.github.com/01org/dpdk-ovs/.

[3] Leading operators create ETSI standards group for network functions virtualization. http://www.etsi.org/index.php/news-events/news/644-2013-01-isg-nfv-created,.

[4] Open vSwitch. http://www.openvswitch.org/.

[5] Snort: An Open Source Intrusion Prevention System. http://www.snort.org/.

[6] Tcpreplay - Pcap editing and replaying utilities. http://tcpreplay.appneta.com/.

[7] The Bro Network Security Monitor. http://www.bro.org/.

[8] A. Belay, G. Prekas, A. Klimovic, S. Grossman, C. Kozyrakis, and E. Bugnion. Ix: A protected dataplane operating system for high throughput and low latency. In *OSDI*. USENIX Association, 2014.

[9] M. Dong, T. Lan, and L. Zhong. Rethink energy accounting with cooperative game theory. In *In MobiCom*. ACM, 2014.

[10] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella. Opennf: Enabling innovation in network function control. In *SIGCOMM*. ACM, 2014.

[11] A. Ghodsi, V. Sekar, M. Zaharia, and I. Stoica. Multi-resource fair queueing for packet processing. In *SIGCOMM*. ACM, 2012.

[12] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: Research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, 2008.

[13] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3), Aug. 2000.

[14] X. Li and C. Qian. Low-complexity multi-resource packet scheduling for network function virtualization. In *INFOCOM*, 2015.

[15] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. Clickos and the art of network function virtualization. In *NSDI*. USENIX Association, 2014.

[16] R. Mijumbi. On the energy efficiency prospects of network function virtualization. *arXiv preprint arXiv:1512.00215*, 2015.

[17] R. Potharaju and N. Jain. Demystifying the dark side of the middle: A field study of middlebox failures in datacenters. In *IMC*. ACM, 2013.

[18] L. Rizzo. Netmap: A novel framework for fast packet i/o. In *ATC*. USENIX Association, 2012.

[19] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. USENIX, 2012.

[20] G. Tang, W. Jiang, Z. Xu, F. Liu, and K. Wu. Zero-cost, fine-grained power monitoring of datacenters using non-intrusive power disaggregation. In *Middleware*. ACM, 2015.

[21] W. Wang, C. Feng, B. Li, and B. Liang. On the fairness-efficiency tradeoff for packet processing with multiple resources. In *CoNEXT*. ACM, 2014.

[22] W. Wu, K. He, and A. Akella. Perfsight: Performance diagnosis for software dataplanes. In *IMC*. ACM, 2015.

[23] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos. Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. *Proceedings of the IEEE*, 102(1):11–31, 2014.

[24] H. Xu and B. Li. Reducing electricity demand charge for data centers with partial execution. In *e-Energy*. ACM, 2014.

[25] Z. Zhou, F. Liu, H. Jin, B. Li, B. Li, and H. Jiang. On arbitrating the power-performance tradeoff in saas clouds. In *INFOCOM*, pages 872–880, April 2013.