This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

Power Consumption Model of NDN-based Multicore Software Router based on Detailed Protocol Analysis

Kaito Ohsugi, *Non-Member,* Junji Takemasa, *Non-Member,* Yuki Koizumi, *Member, IEEE,* Toru Hasegawa, *Member, IEEE,* and Ioannis Psaras, *Member, IEEE*

***Abstract*—NDN (Named Data Networking) has received considerable attention recently, mainly due to its built-in caching, which is expected to enable widespread and transparent operator-controlled caching. One of the important research challenges is to reduce the amount of power consumed by NDN networks as it has been shown that NDN's name prefix matching and caching are power-hungry. As a first step to achieving power-efficient NDN networks, in this paper, we develop a power consumption model of a multicore software NDN router. By applying this model to analyze how caching reduces power, we report that caching can reduce power consumption of an NDN network if the power consumption of routers is in proportion to their load and the computation of caching is as light as that of forwarding.**

***Index Terms*—ICN (Information Centric Networking), NDN (Named Data Networking), Green Network, Power Consumption Model, Multicore Software Router**

I. INTRODUCTION

DUE TO its host-based communication and end-to-end approach, IP cannot naturally provide rich functions such as mobility, multicasting and *in-network* caching. For example, locators of a host change as it moves, while the host name remains unchanged. Since IP routers do not have functions to handle such changing locators, mobility management should be implemented by special severs. In order to overcome such problems inherent to IP, NDN (Named Data Networking) [1] is designed as an alternative Internet architecture. NDN naturally supports the above functions by adopting name-based routing/forwarding and ensures that NDN routers keep state of routes and caches.

Despite the fact that NDN provides numerous functions, time-consuming name prefix matching and caching raise a couple of issues related to forwarding performance [2] and power consumption. Since name prefix matching and caching are time-consuming, high-performance NDN router implementation has become a hot research topic [3], [4], [5].

The research leading to these results has received funding from the EU-JAPAN initiative by the EC Seventh Framework Programme (FP7/2007-2013) Grant Agreement no. 608518 and NICT under Grant Agreement no. 167 (GreenICN project) and UK EPSRC COMIT grant no. EP/K019589/1.

Kaito Ohsugi, Junji Takemasa, Yuki Koizumi, and Toru Hasegawa are with Graduate School of Information Science and Technology, Osaka University, Japan (e-mail: k-ohsugi@ist.osaka-u.ac.jp; j-takemasa@ist.osaka-u.ac.jp; ykoizumi@ist.osaka-u.ac.jp; t-hasegawa@ist.osaka-u.ac.jp).

Ioannis Psaras is with the Electronic and Electrical Engineering Department, University College London, United Kingdom (e-mail: i.psaras@ucl.ac.uk).

NDN router caching reduces the amount of traffic forwarded towards upstream routers and thus, reduces the power consumed by their forwarding devices such as interface cards. A well-studied issue is optimizing cache placement, which determines where in the network to place caches [6], [7] assuming that memory devices for caching consume considerable amounts of power even while in the idle state. For example, Perino et al. assume that DRAM (Dynamic Random Access Memory) devices used for CSs (Content Stores) consume 0.023 W/MB while in the idle state [3]. However, the power they consume is being reduced by lowered voltage and improvements in manufacturing technology. Vogelsang predicts that the decrease in power per bit consumed by DDR SDRAM (Double-Data-Rate Synchronous DRAM) is 1.2 per generation of DDR [8].

We argue that this trend towards lower power consumption implies that *the power consumed by time-consuming name prefix matching and caching will become larger than the power consumed by the memory devices themselves*. This raises the following question: *does caching actually reduce the power consumed by an NDN network by compensating for power-hungry name prefix matching and caching?* Thus, in this paper, we address *the tradeoff relation between the increase of power consumed by name prefix matching and caching of downstream routers and the decrease of power consumed by upstream routers due to traffic reduction.*

As a first step, we model how a PC-based multicore software NDN router consumes power. We chose a software router based on PC-based hardware platform as a target NDN router as recent studies [4], [5] show that well-engineered NDN software routers achieve high-performance name-based forwarding and caching.

In [9], we empirically developed a power consumption model of a PC-based multicore CCNx software router and made two contributions. First, we modelled how individual devices such as a chassis, a CPU device and a NIC (Network Interface Card) consume power under various loads. Second, we empirically analysed how many CPU cycles are needed by each function of the CCNx source code and then calculated the loads of individual devices based on the cycles. This two-step process made it possible to estimate the power consumed by a PC-based CCNx router when its average cache hit rate and chunk size are given. A very important finding of our study in [9] is that *the power consumed by caching is so large that traffic reduction due to caching does not reduce the overall*

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

power consumed by the network.

In this paper, we extend our previous work in [9] to develop a general power consumption model. Despite its initial success, the model in the earlier paper [9] is preliminary and overlooks several important issues. The goal of this paper is to develop a general power consumption model by addressing factors missing from the previous model. The improvements on the previous paper are summarized as follows:

- We model the power consumed by multiple CPU devices using power optimization techniques, whereas in the previous paper these were disabled in order to avoid the difficulty in modeling their non-linear effects. We extend the modeling technique developed in [10] so that the power consumed by multiple CPU devices is estimated. This is an important requirement, given that power consupmtion of CPU devices is proportional to the load [11].
- We extend the study to three PC-based servers, in contrast to our previous paper in [9] where we used only one.
- The model is validated by actually running the NFD (NDN Forwarding Daemon) source code [12] on the two platforms, while such validations were not performed in the previous paper. The power consumed by the platforms running the NFD source code is precisely estimated by the model under the same cache hit rates.
- This paper uses the power consumption model which is developed by using the well-engineered NDN router [4] as a reference; in our previous study, instead, we used the NFD router, which, reportedly, does not achieve high forwarding performance.

The above improvement in accuracy and coverage of multiple server computers enhances the reliability of the observations made based on the experience of developing the model and calculating the power consumed by an NDN network. The contributions of the paper are summarized as follows:

- This paper sheds light on the power consumed by name prefix matching and caching, whereas most studies on power reduction in an NDN network consider power consumed by memory devices while in the idle state [6], [7].
- This paper develops a general method for developing a power consumption model of a PC-based multicore NDN software router by applying the method to the three different server computers.
- The paper precisely models multicore CPU devices with power management states. This makes it clear that power consumption of CPU devices is proportional to their load if they are highly loaded and thus this energy-proportionality is an important requirement to reduce power consumption of a network by caching.
- By using the power consumption model of the well-engineered NDN router, the paper clarifies that caching contributes to reducing power consumed by an overall NDN network if well-engineered NDN routers are used.

The rest of this paper is organized as follows. Section II summarizes the related work. Section III describes the reference architecture. Sections IV and V empirically model how a multicore software router based on NFD [12] consumes power. Section IV models the power consumed by the hardware platform and Section V focuses on the power consumed by NFD packet forwarding. Section VI applies the model to estimate power consumed by networks of three topologies. Section VII concludes the paper.

II. RELATED WORK

Since name prefix matching and caching are time-consuming, high-performance NDN router implementations are a hot research topic. Perino et al. [3] were the first to address this issue and predict future name prefix matching performance. So et al. [4] designed a high-performance forwarding algorithm and implemented it on a commercial router chassis. Focusing on caching, Rossini et al. [5] design a caching algorithm to achieve high performance content access on multi-terabyte caching devices. Whereas these studies focus on individual routers, Fukushima et al. design a protocol which avoids redundant longest prefix matching due to neighboring routers' cooperation [13]. In contrast, this paper focuses on the power consumed by packet-level forwarding.

Many studies focus on the caching functions because traffic reduction would contribute to energy savings. Lee et al. [14], [15] investigated how much power is reduced by reducing hop counts to obtain contents. In their simulations, they used a power consumption model which only considers the power consumed by lower layer packet forwarding devices. Choi et al. [6] show that the power consumed by memory devices used for caching and for the forwarding processes is not insignificant. Imai et al. [7] proposed a method of determining capacities of NDN routers' memory devices so that the power they consume is minimized.

Many power consumption models focus on the power consumed by memory devices because they are power-hungry even while in idle state. In contrast, this paper focuses on the power consumed by packet forwarding, taking into account the trend towards low power consumption by memory devices [8], [16].

Bolla et al. [17], [10] model the power consumed by a CPU device with power optimization techniques by considering two power management states. Although our study is partially motivated by [17], [10], in this paper, we model multiple CPU cores (compared to only one in [17], [10]) and empirically validate the developed model by applying it to two different commercial CPU devices. Finally, we take into account all elements of the router (*e.g.*, chassis, memory devices and CPU), in contrast to [17], [10], where only the CPU is taken into account.

III. REFERENCE ARCHITECTURE

A. Hardware Platform

We selected multicore software routers as our target hardware platforms due to the following reasons. First, full-fledged NDN routers that have all NDN functions would not be used in backbone networks but only in access networks. This is because caching in backbone networks is not as effective as it is in access networks [18]. Second, it is natural that

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

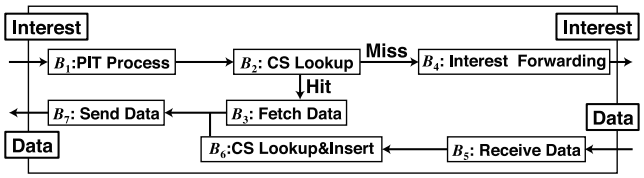


Fig. 1: Flow of Blocks of NFD Source Code

NDN functions are not implemented by interface cards, but by service cards like the ISM (Integrated Service Module) cards in a vendor's routers. So et al. [4] showed that 20 Gbps throughput is feasible on such service cards in commercial routers, which are multicore software routers consisting of a service card and multiple interface cards. In this paper, we use the three PC-based hardware platforms, each of which consists of one CPU device with 4 CPU cores or two CPU devices with 4 CPU cores, one DDR3 memory device, a chassis and one NIC (network interface card). The DDR3 memory device is used to store both tables and packets.

B. NFD Software

1) Packet Processing Flow: We selected NFD [12] as an implementation of NDN. We analyze the NFD source code and group functions into seven groups of functions, each of which is referred to as a block. Figure 1 shows the flow through the seven blocks B_1 to B_7. Blocks B_2, B_3 and B_6 are the blocks used for caching computation. Block B_2 looks up a Data packet requested by an ingress Interest packet. Then block B_3 fetches (reads) the hit Data packet from the CS. Then block B_6 inserts an ingress Data packet into the CS.

The blocks perform request/reply NDN communication using Interest and Data packets as summarized below. First, a requestor of an information piece sends an Interest packet and its destination is specified as the name of the piece, i.e., a human-readable hierarchical name like a URL (universal resource locator). When a router receives the Interest packet (at B_1), the block checks whether the packet is retransmitted or not by checking the PIT (pending interest table). Then B_2 checks whether the Data packet holding the information piece is stored in the CS. If the piece is in the CS, B_3 fetches the Data packet from the CS and B_7 sends back the Data packet to the requestor. Block B_2, for looking up the CS, is computation and memory intensive.

Otherwise, B_4 forwards the Interest packet to an upstream router after performing longest-prefix matching using the FIB (forwarding information table), which holds pairs of name prefixes and outgoing interfaces. In addition, the block creates an entry in the PIT to record the interface that the Interest packet comes from, and thus it is computation intensive.

Second, when the router receives a Data packet at B_5, B_6 stores the Data packet in the CS. At the same time, an unpopular Data packet may be replaced by this Data packet. This block is the most computation and memory intensive among all the blocks, and the memory device consumes a large amount of power. Then B_7 sends the Data packet to the requestor after removing the entry from the PIT.

Interest and Data packets are encapsulated by UDP/IP packets and the NIC forwards the encapsulated packets.

2) Multi-Threading: How to make the NFD source code multi-threaded is important because the hardware platform provides multiple threads, while the NFD source code is single-threaded. In this study, we consider that the NFD code is multi-threaded by making the following assumptions. First, one thread handles sending/receiving packets to/from the NIC and dispatches a received packet to one of the other threads that perform NDN protocol processing. Second, each thread for NDN protocol processing is assigned a CS for storing the information for part of the whole name space so that it does not access CSs of the other threads. In other words, the name spaces of individual threads' CSs are separated from each other. This avoids mutual-exclusion of the threads. Third, the minimum number of threads is greater than the number of CPU cores. The threads are assigned to CPU cores so that the number of active CPU cores is minimized. How many CPU cores are used is determined from the total load of the NDN router.

IV. POWER CONSUMPTION MODEL

We develop a power consumption model of a multicore software NDN router to satisfy the following requirements. The first requirement is that the model should reflect the loads on a hardware platform, that is, the consumed power should be a function of the loads. Such loads include a CPU load, an access rate to DRAM (DDR3) devices, and so forth. The second requirement is that the above-mentioned loads on the hardware platform should be derived from loads on NDN packet forwarding. Sections IV and V address the first and second requirements, respectively.

A. Formulation

We formulate the power consumed by a PC-based hardware platform having the minimum configuration. The power $\phi_{\rm router}$ [W] is defined as,

$$\phi_{\rm router}(c_{\rm ndn}, r_{\rm mem}, \lambda_{\rm ip}) = \phi_{\rm cpu}(c_{\rm ndn}) + \phi_{\rm mem}(r_{\rm mem}) + \phi_{\rm nic}(\lambda_{\rm ip}) + \Phi_{\rm chassis}. \quad (1)$$

It is parameterized by the following three parameters: the load on the CPU device ($c_{\rm ndn}$ [cycle]), the number of bytes accessed in the DDR device per second ($r_{\rm mem}$ [byte/s]), and the IP packet forwarding rate ($\lambda_{\rm ip}$ [packet/s]). To measure the CPU load, we use the CPU cycles incurred for processing tasks, such as NDN packet processing. Each term in Eq. (1), which indicates the power consumed by each hardware component, is explained as follows:

- $\phi_{\rm cpu}(c_{\rm ndn})$ [W] is the power consumed by the CPU device. It is a function of the CPU load $c_{\rm ndn}$.
- $\phi_{\rm mem}(r_{\rm mem})$ [W] is the power consumed by accessing the DDR3 device. It is a function of the number of bytes accessed per second $r_{\rm mem}$.
- $\phi_{\rm nic}(\lambda_{\rm ip})$ [W] is the power consumed by the NIC. It is a function of the IP packet forwarding rate $\lambda_{\rm ip}$.

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

- Φ_{chassis} [W] is the power consumed by the chassis when the router is idle. It includes the power consumption of all devices.

Note that we use capital and small letters to express constants and variables, respectively.

In this section, we empirically measure the above four terms in order to model them. For the measurement, we employ two different kinds of computer architectures: one for general-purpose computing (x86-64), and one for mission-critical computing (IA-64). From the two architectures, we select three server computers: a high-performance server computer for general-purpose computing (server 1), a low-energy server computer for general-purpose computing (server 2), and a high-performance server computer for mission-critical computing (server 3). Server 1 has two Intel® Xeon® E5620 processors (2.4 GHz × 4 cores), one DDR3 12 GB memory device, one Intel® X540-T2 10GBASE-T NIC. Server 2 has one Intel® Xeon® E3-1220 processor (3.10 GHz × 4 cores), one DDR3 16 GB memory device, one Intel® X540-T2 10GBASE-T NIC. Server 3 has one Intel® Itanium® 9520 processor (1.73 GHz × 4 cores), one DDR3 8 GB memory device, and one HP® Integrity Ethernet I/O adapter. To simplify the notation, we refer to Intel® Xeon® E5620, Intel® Xeon® E3-1220, and Intel® Itanium® 9520 processors as E5620, E3-1220, and 9520 processors, respectively hereafter. As operating systems, we use Ubuntu 13.10 for the general-purpose servers and HP-UX 11.31 for the mission-critical server. We use a power meter and a current transformer developed by Omron® (ZN-CTX21 and ZN-CTS51-200As). The power is measured in joules per second (watt). Each measurement is performed for 30 minutes, consisting of a 10-minute warm-up phase and a 20-minute measurement phase. To ensure that we are measuring the servers in their steady state, we put the target load on the servers without taking measurements during the warm-up phase and then measure the power consumption every minute, i.e., we take 20 samples during the measurement phase.

B. Power Consumed by Chassis

We measure the power consumed by the PC under the condition where all the CPU cores are idle and the NIC is connected to an Ethernet switch but no frames are sent or received. The averages of the power consumption of servers 1, 2, and 3 are 68.43, 34.20, and 117.82 [W] and their two-sided 95% confidence intervals are [34.09, 34.30], [68.24, 68.62], and [117.62, 118.02], respectively. Since the confidence intervals are narrow enough, we determine Φ_{chassis} for servers 1, 2, and 3 to be 34.20, 68.90, and 117.82 [W], respectively.

C. Power Consumed by CPU Devices

1) Overview: Modern CPU devices employ several power management states to achieve energy-efficient computing. Therefore, we hypothesize that CPU devices consume power in proportion to their loads if they are highly loaded. To validate the hypothesis, we carefully model several power management states that a modern CPU device employs. Our CPU power consumption model estimates the power consumed by multiple multicore CPU devices by incorporating power management states inside each CPU core and device. We briefly describe the power management states of CPU devices and strategies for controlling the power management states. Then, we build the CPU power consumption model. Finally, we validate it on the basis of empirical measurements and determine the parameters of the model for several commercial CPU devices.

2) CPU Power Management States: Modern CPU devices have two kinds of power management states, low-power idle states (C-states) and performance states (P-states) [19].

The C-states are used to save the power at idle time. When a CPU core is idle, it enters one of idle states, which are numbered from C1 to Cn states. When tasks arrive, it reverts to the active state, which is referred to as the C0 state. A CPU core has several idle states, e.g., C1, C3, and C6 for E5620, E3-1220, and 9520 processors, and a CPU core in a higher numbered C-state requires less power. We model those idle states together and thus simply refer to the idle states as the C1 state, hereafter. In the case of multicore CPU devices, there are several restrictions on controlling C-states. A CPU device has a per-device C1 state, which is a power saving state for the entire CPU device, and each CPU core has a per-core C1 state. Each CPU core can enter its per-core C1 state independently of other CPU cores, whereas an entire CPU device can enter its per-device C1 state only if all its CPU cores are in their per-core C1 states.

In contrast to the C-states, the P-states are used for reducing the power consumption while the CPU device is active, i.e., the C0 state. While a CPU core is working in the C0 state, its power consumption depends on the P-states, which are pairs of the selected operating voltages and frequencies. A CPU device has several P-states numbered from P0 to Pn. For instance, an E5620 processor has seven P-states, P6 to P0, with frequencies ranging from 1.6 to 2.4 GHz. The P0 state is the highest performance state, i.e., the highest operating frequency, voltage, and power consumption. In subsequent P-states, the operating frequency and voltage are progressively reduced in tandem. Note that, in a similar way to the per-device C-states, the P-states are per-device states and thus they affect all CPU cores in a CPU device, i.e., all CPU cores work at the same frequency. The power management using the P-states is also referred to as dynamic voltage and frequency scaling (DVFS).

3) Strategy for Controlling CPU Power Management States: In order to build a power consumption model of multiple multicore CPU devices, we have to model *i)* a strategy for assigning the load incurred for processing NDN packets to multiple CPU devices and cores, and *ii)* a strategy for controlling P-states according to the load on each CPU core. We craft those strategies so that the number of active CPU devices and cores would be minimized.

We assume that a PC platform has N_{cpu} identical CPU devices, each of which has N_{core} identical CPU cores. The CPU devices offer a set of P-states $\mathbf{P}$ and the operating frequency in the P-state p ($p \in \mathbf{P}$) is h_p. We denote the highest performance P-state (P0 state) as p_0, i.e., the highest operating frequency of the CPU device is h_{p_0}. We use CPU cycles per second to measure the CPU load [cycle/s]. Since a CPU core working in the P-state p can process a h_p CPU workload per

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

second [cycle/s], the processing capacity of a CPU core is equivalent to its operating frequency.

Regarding strategy *i)*, the load for processing NDN packets, c_{ndn}, is assigned to CPU devices, so that the number of active CPU devices n_{cpu} is minimized. Since the power consumption of a CPU device increases steeply when any of CPU cores enter their per-core C0 state, which results in the transition of the entire CPU device to its per-device C0 state, minimizing n_{cpu} is a good strategy for minimizing the energy consumed by CPU devices. To minimize n_{cpu}, all CPU cores of the $n_{\mathrm{cpu}}-1$ CPU devices, which have N_{core} CPU cores, must work at their maximum frequency h_{p_0}. Therefore, using a ceiling function, n_{cpu} is expressed as

$$n_{\mathrm{cpu}} = \left\lceil \frac{c_{\mathrm{ndn}}}{N_{\mathrm{core}} h_{p_0}} \right\rceil. \tag{2}$$

In this case, $n_{\mathrm{cpu}}-1$ CPU devices are fully utilized, i.e., all N_{core} CPU cores in the $n_{\mathrm{cpu}}-1$ CPU devices process NDN packets at their maximum frequency h_{p_0}. Therefore, the load processed by those fully utilized $n_{\mathrm{cpu}}-1$ CPU devices is $(n_{\mathrm{cpu}}-1)\cdot N_{\mathrm{core}} h_{p_0}$ and the remainder of the load,

$$c_{\mathrm{device}} = c_{\mathrm{ndn}} - (n_{\mathrm{cpu}}-1)\cdot N_{\mathrm{core}} h_{p_0}, \tag{3}$$

is processed by the least loaded CPU device.

The load c_{device} is assigned to CPU cores in the least loaded CPU device so that the number of active CPU cores is minimized. Since the power consumption of a CPU core is also a concave increasing function of its load, as shown in Section IV-C5, it is preferable to assign as much load as possible to $n_{\mathrm{core}}-1$ CPU cores and the remainder to another one. The number of active CPU cores n_{core} in the least loaded CPU device is expressed as

$$n_{\mathrm{core}} = \left\lceil \frac{c_{\mathrm{device}}}{h_{p_0}} \right\rceil = \left\lceil \frac{c_{\mathrm{ndn}} - (n_{\mathrm{cpu}}-1)\cdot N_{\mathrm{core}} h_{p_0}}{h_{p_0}} \right\rceil. \tag{4}$$

In this case, N_{core} CPU cores in the $n_{\mathrm{cpu}}-1$ CPU devices and $n_{\mathrm{core}}-1$ CPU cores in the least loaded CPU device process NDN packets at their maximum frequency h_{p_0}. Hence, the least loaded CPU core in the least loaded CPU device processes the load c_{core} and it is calculated as

$$\begin{aligned} c_{\mathrm{core}} &= c_{\mathrm{device}} - (n_{\mathrm{core}}-1)\, h_{p_0} \\ &= c_{\mathrm{ndn}} - ((n_{\mathrm{core}}-1) + (n_{\mathrm{cpu}}-1)\cdot N_{\mathrm{core}})\cdot h_{p_0}. \end{aligned} \tag{5}$$

Next, we define a strategy for selecting P-states for the least loaded CPU device according to the CPU load c_{device}. We construct the strategy so that the operating frequency is increased if the utilization of a CPU core, which is defined as the CPU load divided by the processing capacity of the CPU core, exceeds a threshold θ. Such a strategy is often used in the current operating systems. For instance, an advanced configuration and power interface (ACPI) CPU driver employs this strategy. The P-state for the load c_{device} is modeled as,

$$p(c_{\mathrm{device}}) = \begin{cases} p_0 & \text{if } \dfrac{c_{\mathrm{device}}}{h_{p_0}} \geq \theta \\ \mathrm{argmin}_k\, h_k \text{ s.t. } \dfrac{c_{\mathrm{device}}}{h_k} < \theta & \text{otherwise.} \end{cases} \tag{6}$$

For instance, the threshold θ is defined as 0.8 for the "conservative" mode in the ACPI CPU driver embedded in Linux kernels [20]. If the utilization $c_{\mathrm{device}}/h_{p_0}$ exceeds the threshold θ, the P-state of the CPU device is set to p_0. In this case, all CPU cores work at the maximum frequency h_{p_0} since P-states are per-device states.

4) CPU Power Consumption Model: The power consumed by a CPU device ϕ_{cpu} [W] is determined by the load incurred for processing NDN packets c_{ndn} as

$$\begin{aligned} \phi_{\mathrm{cpu}}(c_{\mathrm{ndn}}) = \Phi_{n_{\mathrm{core}},p} \cdot \left(\frac{c_{\mathrm{core}}}{h_p}\right)^{\sigma_{n_{\mathrm{core}},p}} + \sum_{k=1}^{n_{\mathrm{core}}-1} \Phi_{k,p} \\ + \Phi \cdot (n_{\mathrm{cpu}}-1). \end{aligned} \tag{7}$$

The first and second terms on the right hand side are the power consumption of the least loaded CPU device and the third one is that of the fully utilized CPU devices. Since the power consumption of the fully utilized CPU devices, Φ, is constant, the total power consumption of the $(n_{\mathrm{cpu}}-1)$ fully utilized CPU devices is simply derived as $\Phi\cdot(n_{\mathrm{cpu}}-1)$. In contrast, the power consumption of the least loaded CPU device depends on its P-state. Since the P-states are the per-device states, the P-state of the CPU cores in the least loaded CPU device are the same and p is derived by using Eq. (6). The second term is the power consumed by fully utilized CPU cores in the least loaded CPU device. Our empirical measurements show that the maximum power consumption of CPU cores differs slightly depending on the number of active CPU cores though the CPU cores are identical and all of them are operated at the same frequency h_p. Therefore, we model the maximum power consumption of CPU cores as $\Phi_{n_{\mathrm{core}},p}$, which indicates the maximum power consumption of a CPU core in the case that n_{core} CPU cores are active and their P-states are p. To derive the power consumption of the fully utilized CPU cores, we use their sum from $k=1$ to $n_{\mathrm{core}}-1$. Note that if $n_{\mathrm{core}}=1$, the second term is zero, and otherwise p is always p_0 because the P-states are per-device states. Therefore, for deriving the second term in Eq. (7), we need only $\Phi_{n_{\mathrm{core}},p}$ in the case where $p=p_0$. We describe our empirical measurement of $\Phi_{n_{\mathrm{core}},p_0}$ in the following subsection.

Finally, we explain the first term on the right hand side of Eq. (7), which indicates the power consumed by the least loaded CPU core in the least loaded CPU device. This depends on its P-state p and the load c_{core}. Therefore, to derive this, we have to model the C-states, i.e., the power consumption when the CPU cores transit between the C0 and C1 states depending on the load c_{core}, and the P-state, i.e., the power consumption of the CPU core for each P-state while the CPU core is in the C0 state. Our model is based on the power consumption model of a single core CPU device proposed in [10], which models the C-states as $((1-c/h_p)\cdot(\Phi_i^{(p)})^{1/\sigma} + c/h_p\cdot(\Phi_a^{(p)})^{1/\sigma})^{\sigma}$, where $\Phi_i^{(p)}$ and $\Phi_a^{(p)}$ are the power consumption of a CPU core with the P-state p in the C1 and C0 state, respectively. The constant parameter σ determines the shape of the power consumption curve. The case $\sigma=1$ corresponds to the ideal case, where there is no overhead when entering or exiting the idle state. In reality, a latency exists when reverting from the C1 state to the C0 state and a certain amount of power is

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

TABLE I: Parameters of $\phi_{\rm cpu}(c_{\rm ndn})$ for E5620 Processor

$n_{\rm core}$	h_p [GHz]	ν	$\Phi_{\rm a\ min}$	$\sigma_{n_{\rm core},p}$	$\Phi_{n_{\rm core},p}$
1	1.6	$1.35 \cdot 10^3$	16.86	0.30	24.33
	1.73			0.13	25.08
	1.87			0.15	25.88
	2.0			0.065	26.66
	2.13			0.14	27.49
	2.27			0.074	28.36
	2.4			0.12	29.22
2	2.4	-	-	0.84	5.94
3	2.4	-	-	0.74	5.37
4	2.4	-	-	0.81	6.51

TABLE II: Parameters of $\phi_{\rm cpu}(c_{\rm ndn})$ for E3-1220 Processor

$n_{\rm core}$	h_p [GHz]	ν	$\Phi_{\rm a\ min}$	$\sigma_{n_{\rm core},p}$	$\Phi_{n_{\rm core},p}$
1	1.6	$1.89 \cdot 10^3$	3.63	0.46	6.25
	1.8			0.40	6.69
	2.0			0.32	7.16
	2.2			0.38	7.66
	2.4			0.36	8.20
	2.6			0.40	8.78
	2.8			0.38	9.39
	3.1			0.48	10.40
2	3.1	-	-	0.59	5.71
3	3.1	-	-	0.69	6.86
4	3.1	-	-	0.56	5.24

TABLE III: Parameters of $\phi_{\rm cpu}(c_{\rm ndn})$ for 9520 Processor

$n_{\rm core}$	h_p [GHz]	ν	$\Phi_{\rm a\ min}$	$\sigma_{n_{\rm core},p}$	$\Phi_{n_{\rm core},p}$
1	1.066	2.58	0.00	0.68	2.23
	1.6			0.78	6.35
	1.733			0.34	7.80
2	1.733	-	-	0.55	3.92
3	1.733	-	-	0.74	1.39
4	1.733	-	-	0.34	1.21

consumed during the transition. Thus, the power consumption is an increasing concave function of the load λ, i.e., the parameter σ for a current CPU device satisfies $0 < \sigma < 1$. To model the P-states, i.e., the maximum power consumption $\Phi_a^{(p)}$ in the case of the P-state p, this model uses the following equation, $\Phi_a^{(p)} = (\Phi_{\rm amin}^{1/\nu} + (\Phi_{\rm amax}^{1/\nu} - \Phi_{\rm amin}^{1/\nu}) \cdot h_p/h_{p_0})^\nu$, where $\Phi_{\rm amax}$ and $\Phi_{\rm amin}$ are the maximum and minimum power consumptions in the C0 state. The constant parameter ν determines the shape of the power consumption curve. The case $\nu = 1$ corresponds to dynamic frequency scaling and $\nu = 2$ corresponds to dynamic voltage scaling. For DVFS, ν is greater than 2.

We extend this model to multiple multicore CPU devices assuming that the CPU power management is performed as discussed in the previous subsection. We set $\Phi_i^{(p)}$ to 0 since the power consumption of the current CPU's idle cores is near-zero independent of other cores [21]. Then, we apply the model [10] to the least loaded CPU core. The first term of Eq. (7) is based on the C-state model in [10]. According to the observations found in our empirical measurements below, we define the parameters $\sigma_{n_{\rm core},p}$ for each P-state p and the number of active CPU cores $n_{\rm core}$ instead of using a single parameter σ. The constant $\Phi_{n,p}$ is the maximum power consumption of the n-th CPU core for a given P-state p in the case that $n-1$ CPU cores are fully utilized. $\Phi_{n_{\rm core},p}$ [W] can be derived as

$$\Phi_{n_{\rm core},p} = \left(\Phi_{\rm a\ min}^{\frac{1}{\nu}} + \left((\Phi_{n_{\rm core},p_0})^{\frac{1}{\nu}} - \Phi_{\rm a\ min}^{\frac{1}{\nu}} \right) \cdot \frac{h_p}{h_{p_0}} \right)^\nu, \tag{8}$$

where $\Phi_{\rm a\ min}$ is the minimum power consumption of the CPU core in the C0 state. The constant ν is a parameter to determine the maximum power consumption of the CPU core for each P-state. Eq. (8) is used to derive $\Phi_{n,p}$ since P-state adaptation is applied only when the number of active CPU cores is one, we apply Eq. (8) for the case of $n_{\rm core} = 1$ and otherwise we use the empirically measured value for Φ_{n,p_0}.

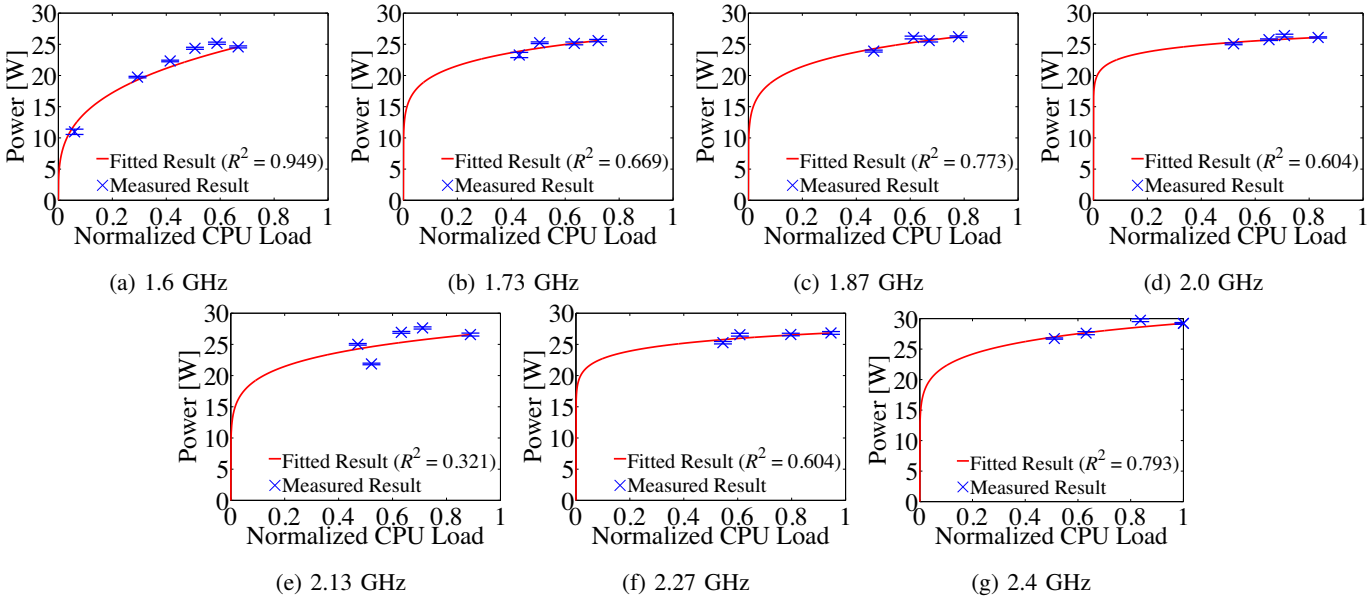
5) Validation and Parameter Fitting: To derive the constants in Eqs. (7) and (8), we measure the power consumption by running a program that performs an infinite loop containing only arithmetic operations and a sleep operation to avoid access to any hardware devices other than the CPU. We vary the CPU load by changing the sleep time. The derived parameters are summarized in Tables I, II, and III.

First, to derive the parameters $\sigma_{n_{\rm core},p}$ and $\Phi_{n_{\rm core},p}$ in the case of $n_{\rm core} = 1$, we measure the CPU power consumption for each P-state by changing the CPU load. The measured and estimated power consumption of the E5620 processor (server 1) for each operating frequency (P-state) is shown in Fig. 2. The horizontal axis shows the offered CPU load normalized to the maximum processing capacity and the vertical axis shows the power consumption. Cross markers and error bars indicate the mean of 20 measured results and 95% confidence intervals. Since the confidence intervals are considerably small for our measurements, as shown in Fig. 2, the confidence intervals for the CPU power measurements are omitted, hereafter. We derive $\sigma_{n_{\rm core},p}$ with least squares approximation. The significance of the regression is measured by the coefficient of determination, $R^2 = 1 - \sum_i (y_i - m_i) / \sum_i (y_i - \bar{y})$, where y_i denotes the sample value with mean $\bar{y}$ whereas m_i is the modeled value, and R^2 is shown in each graph. Due to space limitations, we omit graphs for the E3-1220 and 9520 processors. The tendencies of the fitting results of the E3-1220 and 9520 processors are the same as that of the E5620 processor. R^2 of the fitting results for the E3-1220 processor in the case of h_p = 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8, and 3.1 GHz are 0.934, 0.946, 0.856, 0.922, 0.978, 0.981, 0.945, and 0.954, respectively. Those for the 9520 processor in the case of h_p = 1.066, 1.6, and 1.73 GHz are 0.911, 0.899, and 0.919, respectively. Since R^2 of the fitting results are high, we determine the parameters $\sigma_{n_{\rm core},p}$ and $\Phi_{n_{\rm core},p}$ in the case of $n_{\rm core} = 1$, as shown in Tables I, II, and III.

Next, to derive $\sigma_{n_{\rm core},p}$ in the case that $n_{\rm core}$ is more than 1, we measure the power consumption of the n-th CPU core by changing the load on the core. During the measurement of the n-th CPU core, the other $n-1$ CPU cores are fully utilized and the remaining CPU cores are kept idle. Since the P-state of all CPU cores is p_0 when more than one CPU core is active, we measure the power consumption in the case of the P-state p_0. The measured and fitted results for the E5620 are shown in Fig. 3. The meanings of the axes and markers are the same as those in Fig. 2. R^2 of the fitting results for the E3-1220 processor in the case of $n_{\rm core}$ = 2, 3, and 4 are 0.891, 0.870, and 0.699, and those for the 9520 processor are 0.929, 0.998, and 0.913.

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

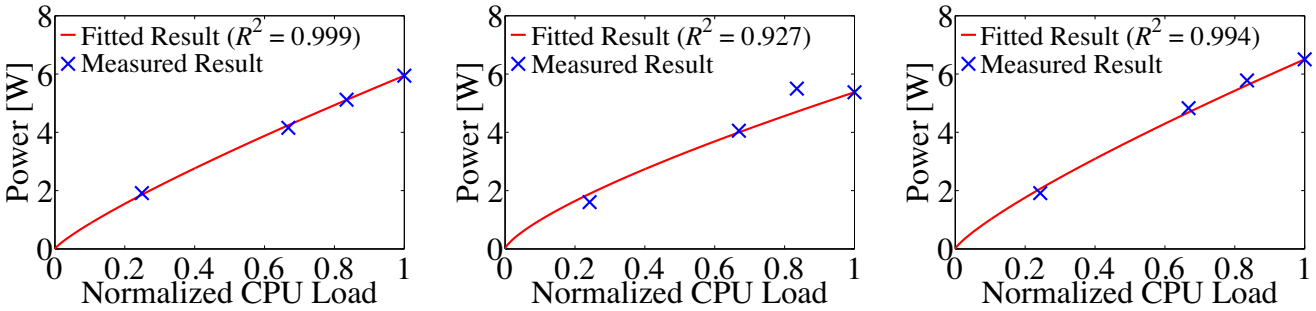
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications



(a) 1.6 GHz (b) 1.73 GHz (c) 1.87 GHz (d) 2.0 GHz

(e) 2.13 GHz (f) 2.27 GHz (g) 2.4 GHz

Fig. 2: Power Consumed by CPU (E5620) Cores at Each P-state



(a) $n_{\text{core}} = 2$ (b) $n_{\text{core}} = 3$ (c) $n_{\text{core}} = 4$

Fig. 3: Power Consumed by CPU (E5620) in the Case of $n_{\text{core}} \geq 2$

To derive the parameters ν and $\Phi_{\text{a min}}$, we measure the power consumption for each P-state when a CPU core is fully utilized and the other CPU cores are kept idle. The measured power and the fitted model are shown in Fig. 4. The horizontal axis shows the operating frequency normalized to the maximum frequency of each CPU core. R^2 of each regression is shown in each graph. Though the error between fitted and measured power consumption of the E5620 processor with operating frequency of 2.13 and 2.27 GHz is large, R^2 is high. R^2 of the fitting results for the E3-1220 and 9520 processors are 0.951 and 0.995, respectively. Therefore, we derive ν for the E5620, E3-1220, and 9520 processors as $1.35 \cdot 10^3$, $1.89 \cdot 10^3$, and 4.22 and $\Phi_{\text{a min}}$ for the E5620, E3-1220, and 9520 processors as 16.86, 3.63, and 0.00, respectively.

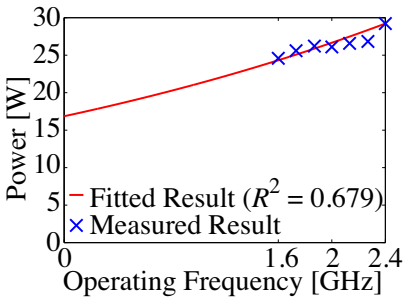
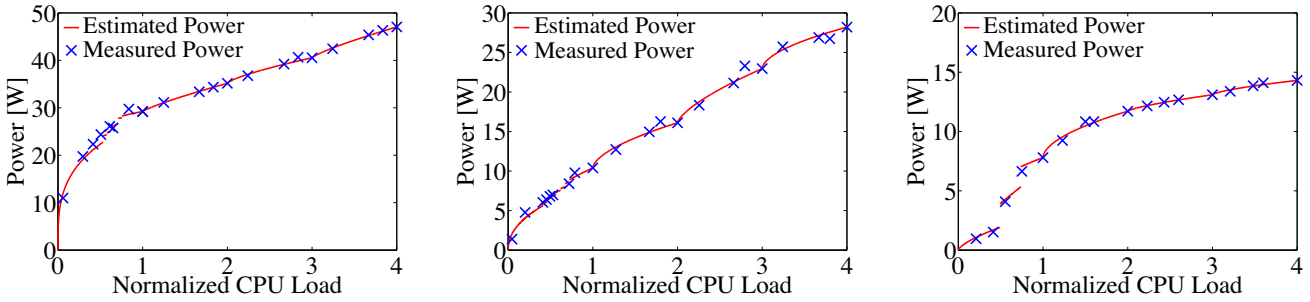


Fig. 4: Maximum Power Consumed by one CPU Core (E5620) in Each P-state

Finally, we show the CPU power consumption of the E5620, E3-1220, and 9520 processors as estimated by our model in Fig. 5. The horizontal axis shows the offered CPU load normalized to the maximum processing capacity, c_{ndn}/h_{p_0}, of each CPU core and the vertical axis shows the power consumption. The significance of the estimation has been shown as the coefficients of determination, R^2, of the fitting results in Figs. 2 to 4. Since modern CPU devices employ several power management states, they are almost energy-proportional to their loads in the case that two or more CPU cores are active. That is, the power consumption of the CPU devices are approximately modeled as a linear function to their loads. We derive a coefficient and an intercept of the linear function by using linear regression. The power consumption of the E5620 processor is approximately modeled

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications



(a) E5620 Processor (b) E3-1220 Processor (c) 9520 Processor

Fig. 5: Power Consumption by CPU Device

as $\phi_{\rm cpu}(c_{\rm ndn}) = 5.88{\cdot}c_{\rm ndn}/h_{p_0}{+}23.58$ and R^2 is 0.997. In the same way, the power consumption of the E3-1220 and 9520 processors are modeled as $\phi_{\rm cpu}(c_{\rm ndn}) = 5.85{\cdot}c_{\rm ndn}/h_{p_0}{+}5.43$ with $R^2 = 0.979$ and $\phi_{\rm cpu}(c_{\rm ndn}) = 1.65{\cdot}c_{\rm ndn}/h_{p_0}{+}8.15$ with $R^2 = 0.943$, respectively.

D. Power Consumed by Memory Device

We formulate the power consumed by accessing data in the DDR3 device $\phi_{\rm mem}(r_{\rm mem})$ [W] as a function of the average number of bytes accessed per second $r_{\rm mem}$ [byte/s] as follows:

$$\phi_{\rm mem}(r_{\rm mem}) = \Psi_{\rm mem} \cdot r_{\rm mem}, \tag{9}$$

where $\Psi_{\rm mem}$ is the energy [joule] consumed to read/write one byte of data from/to the DDR3 device [joule/byte]. Since currently available DDR3 devices do not have DVFS, their power consumption is proportional to the access rate to the devices [22]. Hence, we model the power consumption of the DDR3 device as a linear function to the access rate $r_{\rm mem}$. Note that the power consumed constantly by the DDR3 device, such as the power for refreshing registers, is included in the power consumed by the chassis $\Phi_{\rm chassis}$.

To validate that the power consumed by accessing the DDR3 device is proportional to the rate of accessing it and to derive the constant $\Psi_{\rm mem}$ in Eq. (9), we run a program that repeatedly reads 8 byte data from the array allocated by the malloc function. In general, the DDR3 power consumption for writing and reading data is slightly different but the difference is small. For instance, writing a page to a DDR3 device operating at 1333 MHz consumes 61 nano-joule while reading does 56 nano-joule [22]. Since we measure the power consumed for reading the DDR3 device and estimate $\Psi_{\rm mem}$ based on the measured results, our model may underestimate the power consumed by the DDR3 device. However, the error is at most about 8% according to [22]. Furthermore, the DDR3 device accounts for the small proportion of the total power consumption of the servers. Therefore, the error might be much smaller. The average number of bytes accessed in the DDR3 device per second is measured by the Intel® Performance Counter Monitor. Since all three servers 1, 2, and 3 have DDR3 devices, we use server 2 for this measurement. We derive the power consumed by accessing the DDR3 device by subtracting the power consumed by one active CPU core (10.40 [W]) from the measured power.

Figure 6 shows the power consumed by the DDR3 device at various byte access rates [byte/s]. We derive the constants $\Psi_{\rm mem}$ by using the least squares approximation. Since the confidence intervals of each measured result are small and R^2 is close to 1, we decide on $\Psi_{\rm mem}$ to be $0.61{\cdot}10^{-9}$ [joule/byte].

E. Power Consumed by NIC

We formulate the power consumed by the NIC $\phi_{\rm nic}(\lambda_{\rm ip})$ [W] as a function of the IP packet forwarding rate $\lambda_{\rm ip}$ [packet/s] as

$$\phi_{\rm nic}(\lambda_{\rm ip}) = \Psi_{\rm nic} \cdot \lambda_{\rm ip}, \tag{10}$$

where $\Psi_{\rm nic}$ is the energy [joule] consumed by the NIC for forwarding one IP packet [joule/packet]. Though the power consumed by a currently available NIC may not be energy-proportional, its power consumption in general is much smaller than that of a CPU device [23]. Even though we model it as a linear function to its load, the error between the estimated and measured power consumption might be small enough, which is validated by our empirical measurements below.

We measure the power consumed by the NIC at various rates in the following way. The three PCs are connected by 10 Gbps Ethernet links. One PC is used as an IP router and the other two act as a client and server, respectively. The client sends UDP packets at various rates by running a simple program that switches between sending a UDP packet and sleeping. We measure the power by choosing 1500 bytes as the size of the IP packets.

Figure 7 shows the power consumed by the NIC. The horizontal axis shows the packet forwarding rate [packet/s]. Cross markers and error bars indicate the mean and 95% confidence intervals of measured results. We derive the constant $\Psi_{\rm nic}$ as $1.26 \cdot 10^{-5}$ [joule/packet] using the least squares approximation. The power does not appear to be exactly proportional to the IP packet forwarding rate since the power consumed by the NIC is small and thus the fluctuations in the measured values become relatively large. However, we assume that the power is proportional to the forwarding rate $\lambda_{\rm ip}$. This is because the confidence intervals of each measured result are small and R^2 is close to one, and thus errors between the actual and estimated values would be negligible.

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

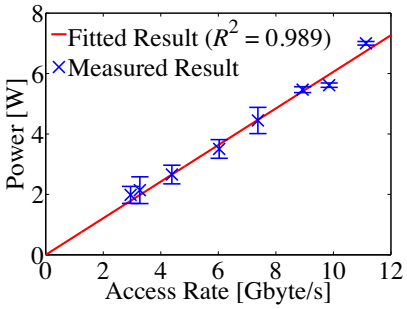


Fig. 6: Power Consumed by DDR3 Device

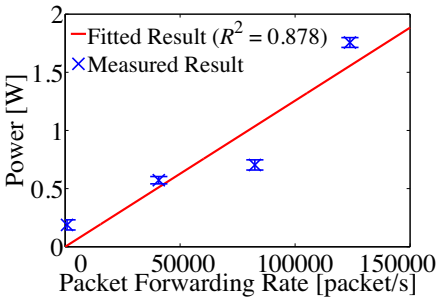


Fig. 7: Power Consumed by NIC

V. PACKET FORWARDING ANALYSIS

This section describes the method of calculating values of the three parameters $c_{\rm ndn}$, $r_{\rm mem}$ and $\lambda_{\rm ip}$ in order to derive the average power consumed by a PC-based multicore NDN software router. The rest of this section describes how average values of the three parameters are calculated from the average input Interest packet rate $\lambda_{\rm ndn}$ [packet/s] and cache hit rate $\pi_{\rm hit}$ of the NDN router.

A. IP Packet Forwarding Rate

The average IP packet forwarding rate $\lambda_{\rm ip}$ [packet/s] is calculated by the following function of $\lambda_{\rm ndn}$ and $\pi_{\rm hit}$:

$$\lambda_{\rm ip}(\lambda_{\rm ndn}, \pi_{\rm hit}) = \lambda_{\rm ndn} \cdot (2 - \pi_{\rm hit}). \tag{11}$$

$\lambda_{\rm ip}$ is calculated assuming that Interest and Data packets are encapsulated by IP packets. Since Interest and Data packets are one-for-one and their flow balances are strictly maintained in NDN [24], we can calculate the average number of Interest and Data packets received and sent per second as follows. The router receives $\lambda_{\rm ndn}$ Interest packets from downstream routers per second. $\lambda_{\rm ndn} \cdot \pi_{\rm hit}$ Data packets are sent back to them per second when Interest packets hit those in the CS. Otherwise, $\lambda_{\rm ndn} \cdot (1 - \pi_{\rm hit})$ Interest packets are sent (forwarded) to upstream routers per second. Then $\lambda_{\rm ndn} \cdot (1 - \pi_{\rm hit})$ Data packets are received from the upstream routers and then they are sent back to the downstream routers per second. Since either an Interest or a Data packet is encapsulated by an IP packet, $\lambda_{\rm ndn} \cdot (2 - \pi_{\rm hit})$ IP packets are sent and the same number of IP packets are received per second. It means that $\lambda_{\rm ndn} \cdot (2 - \pi_{\rm hit})$ IP packets are forwarded. Thus, $\lambda_{\rm ip}(\lambda_{\rm ndn}, \pi_{\rm hit})$ is $\lambda_{\rm ndn} \cdot (2 - \pi_{\rm hit})$.

TABLE IV: Memory Access Ratio

$\pi_{\rm hit}$	0.00	0.05	0.10	0.15	0.20	0.25
$\Gamma_{\rm mem}$	35.67	33.77	32.72	32.23	30.27	28.63

B. Access Rate to DDR3 Device in Bytes

The average power consumed by the DDR3 device $r_{\rm mem}(\lambda_{\rm ndn}, \pi_{\rm hit})$ [byte/s] is calculated by the following function of $\lambda_{\rm ndn}$ and $\pi_{\rm hit}$:

$$r_{\rm mem}(\lambda_{\rm ndn}, \pi_{\rm hit}) = \lambda_{\rm ndn} \cdot S_{\rm ndn} \cdot \Gamma_{\rm mem}. \tag{12}$$

This subsection describes how $r_{\rm mem}(\lambda_{\rm ndn}, \pi_{\rm hit})$, i.e., the average number of bytes accessed in the DDR3 per second, is derived. Since it is difficult to precisely calculate how many bytes the NFD code accesses in the DDR3 device at run time, we estimate it in the following way: first, we measure how many bytes are accessed in the DDR3 device by observing a communicating NFD router. We derive the ratio of the number of accessed (read or written) bytes in the DDR3 to the bytes of information pieces that are retrieved. $\Gamma_{\rm mem}$ in Eq. (12) is this ratio. For example, if $\Gamma_{\rm mem}$ is 10, when a 1 GB of information is retrieved, the NFD router is assumed to access 10 GBs in the DDR3 device.

Second, we assume that the above ratio is always the same for any NFD packet. Here, since $\lambda_{\rm ndn} \cdot S_{\rm ndn}$ is the average number of bytes retrieved in information pieces, Eq. (12) estimates the number of bytes per second accessed in the DDR3 device. We measure the average number of bytes per second accessed in the DDR3 device during the experiments in Section V-B.

We measure $\Gamma_{\rm mem}$ when $\pi_{\rm hit}$ are 0.00, 0.05, 0.10, 0.15, 0.20 and 0.25. When an Interest packet hits a Data packet in the CS, the CS is accessed only once for lookup the corresponding Data packet, but when an Interest packet does not hit any packet in the CS, the CS is accessed twice for the Data packet lookup and insertion. Thus, as shown in table IV, $\Gamma_{\rm mem}$ is small in highly $\pi_{\rm hit}$ since CS is stored in the DDR3 device in NFD.

C. CPU Cycles

The average power consumed by the CPU device is determined by the average number of CPU cycles. This subsection describes how to calculate $c_{\rm ndn}(\lambda_{\rm ndn}, \pi_{\rm hit})$ as a function of $\lambda_{\rm ndn}$ and $\pi_{\rm hit}$. First, we classify all the blocks in the NFD source code described in Fig. 1 into the following three groups of blocks so that the average number of CPU cycles is calculated from the cache hit rate of the router:

- The group of blocks F_1: The block is always run when an Interest packet is received. This group includes the blocks B_1, B_2 and B_7.
- The group of blocks F_2: The block is run only when an Interest packet hits a Data packet contained in the CS. This group includes the block B_3.
- The group of blocks F_3: The block is run only when an Interest packet does not hit any Data packet contained in the CS. This group includes the blocks B_4, B_5 and B_6.

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

TABLE V: The Numbers of Average CPU Cycles

Block	B_1	B_2	B_3	B_4	B_5	B_6	B_7
CPU Cycles	21,426	19,147	3,868	20,443	16,398	83,198	7,973

Thus, the individual terms $\sum_{f\in F_1} C_f$, $\sum_{f\in F_2} C_f$, and $\sum_{f\in F_3} C_f$, where C_f is the average number of CPU cycles to execute a block f, are totals of the CPU cycles consumed to execute all the blocks in the group for processing one pair of an Interest packet and the corresponding Data packet. Since an Interest packet corresponds one-to-one with a Data packet, $\sum_{f\in F_1} C_f + \pi_{\rm hit}\cdot\sum_{f\in F_2} C_f + (1-\pi_{\rm hit})\cdot\sum_{f\in F_3} C_f$ calculates the average CPU cycles with the cache hit rate $\pi_{\rm hit}$ when one Interest packet is received. Therefore, the CPU cycles incurred for processing NDN packets per second $c_{\rm ndn}(\lambda_{\rm ndn}, \pi_{\rm hit})$ [cycle/s] is expressed as

$$c_{\rm ndn}(\lambda_{\rm ndn}, \pi_{\rm hit}) = \lambda_{\rm ndn}\cdot\left(\sum_{f\in F_1} C_f + \pi_{\rm hit}\cdot\sum_{f\in F_2} C_f + (1-\pi_{\rm hit})\cdot\sum_{f\in F_3} C_f\right). \quad (13)$$

D. CPU Cycle Measurements

We measure the average number of CPU cycles of each group as follows. We connect three PCs via a 10-Gigabit Ethernet switch. One PC acts as an NFD router and two PCs act as a client and a server, respectively. The parameters at the NFD layer are chosen as follows. The length of the content name is 9 characters and the number of name components is 2. The size of each chunk is 1 KB.

To measure the CPU cycles required for each block, i.e., B_1 to B_7, the NFD software on the router is run by adding an RDTSC (Read Time-Stamp Counter) instruction, which reads a time stamp counter, a register incremented by CPU cycles in order to measure how many CPU cycles each group consumes. Table V shows the average CPU cycles of the 7 blocks.

E. Estimation Accuracy for One CPU Core

To validate the combination of the equations developed by Sections IV and V, each of which have been already validated in the previous sections, we compare the power estimated by the model and that actually measured by running the NFD code on the PC-based platform at various Interest packet rates. Since the NFD code does not support HP-UX (server 3), we use servers 1 (E5620 processor) and server 2 (E3-1220 processor) in the following experiments. During the measurement, one CPU core is used because the NFD code is single-threaded. The measurements are performed under the same conditions as those in Section IV. The model slightly underestimates the power consumption for both servers 1 and 2 because the model does not incorporate the power consumption other than for the NFD code. That is, several essential programs for operating Linux, such as the kernel and daemons, consume a certain amount of power, which is not included in the estimated power consumption. Although the model underestimates the power consumption, the errors between the measured and estimated power stay within a reasonable range for both servers 1 and 2. We measure the significance of the model by the root mean square error (RMSE). The RMSE values for server 1 in the case of $\pi_{\rm hit}$ 0, 0.1, and 0.2 are 7.75, 8.74, and 8.32, respectively. The RMSE values for server 2 in the case of $\pi_{\rm hit}$ 0, 0.1, and 0.2 are 3.50, 3.33, and 2.68.

VI. CASE STUDY

A. Power Consumption Model of Well-engineered NDN Router

In order to analyze power consumption in commercial networks, in this paper, we select the well-engineered NDN router [4] and develop its power consumption model. Since the NDN source code runs on a PC-based card that has an Intel® CPU processor, we estimate the number of the CPU cycles of its function blocks assuming that it runs on the PC-based hardware platform which this paper models. The numbers are calculated from the numbers reported in [4].

The CPU cycle numbers of processing an Interest packet are calculated as follows: first, the total number of processing an Interest packet is calculated as 2438.68 from the average Interest packet forwarding rate. Second, the numbers of PIT Process B_1 and CS Lookup B_2 are calculated as 421.96 as follows: the well-engineered NDN code uses Siphash as the hash function and the number of CPU cycles of lookup for this hash table is 421.96 [4]. Assuming that the computation of looking up the hash table for the PIT and CS accounts for a large portion of the total computation, we decide the CPU cycle number of the blocks B_1 and B_2 to be 421.96. Finally, the CPU cycle number of Interest Forwarding B_4 is calculated by subtracting the sum of CPU cycle numbers of B_1 and B_2 from 2438.68.

The CPU cycle numbers of processing a Data packet is as follows: first, the total number of processing a Data packet is calculated as 995.90 from the average Interest packet forwarding rate. Second, the CPU cycle number of Receive Data B_5 and CS Lookup & Insert B_6 is assumed to be 421.96 because the blocks look up once the hash tables for the PIT and the CS. Third, the CPU cycle number of Fetch Data B_3 is assumed to be 0 because no hash computation is needed for this block. Finally, the CPU cycle number of Send Data B_7 is calculated as 151.98 by subtracting the CPU cycle numbers of B_5 and B_6 from 995.90.

B. Scenario

This subsection describes the scenario of the case study. Three network topologies are used. The first topology is a line topology of three routers, where the most upstream router is connected by an information server and the most downstream router is connected by a client of sending Interest packets. We use the high-performance server computer with two E5620 processors as the hardware platform for all three routers in this topology. The second network topology is a tree topology, consisting of a root router, which is connected by an information server, and 2 middle routers and 2×4 edge routers. The tree topology is chosen because current

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

ISP (internet service provider) access networks employ a tree-based topology and because the NDN caching functions reduce the amount of traffic more effectively in access networks than core networks [18]. We use the low-energy server with one E3-1220 processor for the edge routers and the high-performance server with two E5620 processors for the middle and root routers since the middle and root routers have to forward more traffic than the edge routers. The third network topology is the Abilene topology, which consists of 9 routers and is an example of a core network. We use the high-performance server with two E5620 processors for all routers in the Abilene topology.

Since video traffic accounts for a large portion of the current Internet traffic, 10 MB is selected as the size of an information piece stored at the server based on a recent study by Zhou et al., which estimates that there are currently $5 \cdot 10^8$ YouTube videos with an average size 10 MB [25]. The number of information pieces G is 160,000 because the Web access logs measured in a medium-size city in the U.S. [26] show that the average number of Web pages requested in a day is about 160,000. Each information piece is divided into chunks with the size of 1024 bytes.

The number of CSs of each router is determined depending on the number of CPU cores M. The CS of each router is divided into equal sized M independent CSs. To simplify the notation, we will refer to the divided CS as sub-CS, hereafter. Each sub-CS is individually assigned to each CPU core. This means that the number of threads is M as well. The total size of the CS is changed from 1 GB to 256 GB.

Whole name space of all the information pieces is divided into equal sized M name spaces and each name space is assigned to one of the sub-CSs. Then, we make two assumptions so that information pieces of the same popularity distribution are handled by M independent caching algorithms. One assumption is that the popularity of the information pieces at the individual sub-CSs, i.e., in the individual name spaces, are the same. The popularity of information pieces in each name space follows the Zipf distribution with the parameter $\alpha = 0.8$ [25]. The other assumption is that caching algorithms are independently executed under the IRM (independent reference model) for sub-CSs and thus requests for information pieces arrive independently at individual sub-CSs according to independent Poisson processes with mean rate $\lambda_m = \lambda/M$, where λ is the mean rate of requests for all information pieces which arrive at the router.

We calculate the cache hit rates of routers in the three topologies on the basis of a model proposed by Martina et al. [27], which analyzes the cache hit rate of each information piece for a FIFO cache under the IRM. By using the model, we derive the cache hit rate of the k-th popular information piece at each sub-CS $\pi_{m,k}$. Then, the expected cache hit rate of the sub-CS π_m is derived from the mean of $\pi_{m,k}$ weighted by the arrival rate of each information piece, $\lambda_{m,k}$. In the same way, the expected cache hit rate of the CS π is derived from the mean of π_m weighted by its arrival rate λ_m. To derive the cache hit rates of upstream routers, we assume that the forwarding process of the k-th popular information piece of each caching algorithm toward an upstream router is also an independent Poisson process with a mean rate of $\omega_{m,k} = \lambda_{m,k} \cdot (1 - \pi_{m,k})$. The mean arrival rate of the k-th popular information piece at an upstream router is simply derived from the sum of the arrival rate of forwarded processes from downstream routers $\omega_{m,k}$. Therefore, we can calculate the cache hit rates of upstream routers by applying the model in [27].

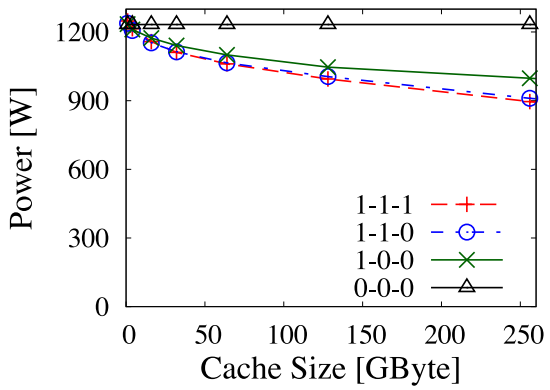


Fig. 8: Power Consumption in Line Topology

C. Power Reduction Analysis

This section analyzes the power consumption of networks under the following conditions. First, in the case of the line and tree topologies, cache functions are placed as follows:

- *case1-1-1*: Cache functions are placed at all (1st, 2nd and 3rd) routers.
- *case1-1-0*: Cache functions are placed at the edge (1st) and intermediate (2nd) routers.
- *case1-0-0*: Cache functions are placed at only the edge (1st) routers.
- *case0-0-0*: Cache functions are placed at none of the routers.

Second, the CS size is changed from 1 GB to 256 GB for the line topology and the CS size of the tree topology is 64 GB. Third, the average Interest packet rate to the edge router is 5,493,165 [packet/s] for the line topology. This corresponds to 45 Gbps of the information retrieval rate from the information server. The number of active CPU cores of the three routers is 8. In contrast, that to the edge routers is changed for the tree topology so that the average Interest packet rate to the root router ranges from 0 to 5,493,165 [packet/s].

Figures 8 and 9 show the power consumption and the cache hit rates of the three routers in the line topology, respectively. In contrast, Fig. 10 shows the power consumption in the tree topology when the CS size is 64 GB and the average Interest packet rate is changing.

Next, we focus on a general core network topology, using the Abilene topology as an example. An information server is placed at Atlanta and a client is connected to a router in every other city. The interest packet rate (packet/s) of each client is determined according to the population of the city. The total information retrieval rate is from 0 to about 45 Gbps. The other conditions are the same as those of the analysis in the tree topology. Figure 11 shows the power both when

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

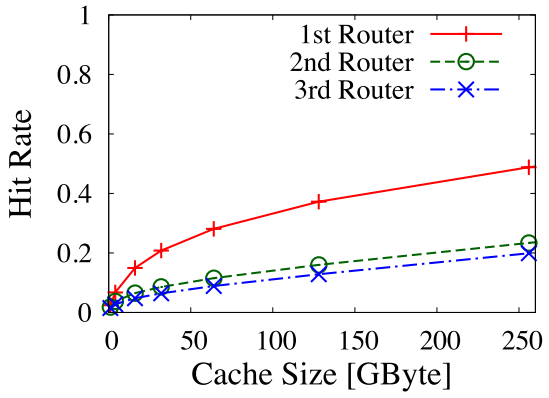


Fig. 9: Cache Hit Rates in Line Topology

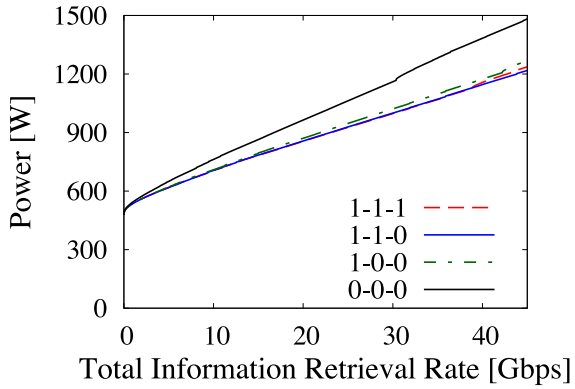


Fig. 10: Power Consumption in Tree Topology

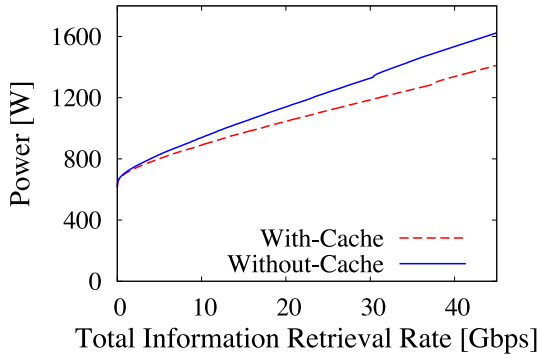


Fig. 11: Power Consumption in Abilene Topology

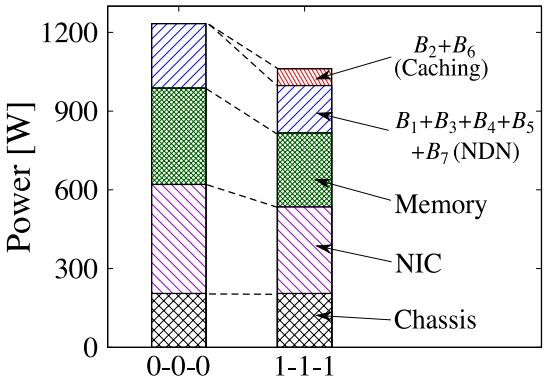


Fig. 12: Power Consumption of Devices with and without Caching

cache functions are placed at all the routers (With-Cache) and at none of the routers (Without-Cache). Caching reduces the power consumed by the Abilene network even if the network is not a simple hierarchical topology.

We note the following observations about the above figures. First, caching reduces the power consumption compared with that without caching in all the topologies. For a more detailed analysis, we analyze how the devices consume power and how the blocks for caching and forwarding consume power at the CPU device. Figure 12 shows the power of the above devices and the above functions in the case that the Interest packet rate of each client is 5,493,165 [packet/s] (45 Gbps) for the line topology and the CS size is 64 GB.

This figure helps us understand the tradeoff relation. The positive effects are two-fold: First, the traffic reduction due to caching reduces the power consumed by the NIC and memory devices of upstream routers. As shown in Fig. 12, the total power consumption of these devices is reduced. Second, the number of CPU cycles consumed by the blocks B_4 and B_5 of the CPU devices of all routers is reduced because the number of Interest packets is reduced. In contrast, a negative effect is that all routers consume CPU cycles of the blocks B_2 and B_6 for caching. It is clear that the power reduction due to the positive effects is larger than that of the power increase due to the negative effects.

Second, the cache hit rates of the 2nd and 3rd routers are still several percent and thus the power is reduced when cache functions are placed at upstream routers. However, the power reduction caused by cache functions at the upstream routers is not remarkable. Thus, it is considered that placing caches only at edge routers is enough for simple networks such 3 level line and tree topologies.

Third, the amount of power reduction compared with the total power is roughly in proportion to the load on the network, i.e., the total packet rate, as shown in Fig. 10. The ratio increases when an NDN network is becoming highly loaded.

D. Lessons Learned

Important lessons are learned from the observations.

An important requirement for power reduction is that the power consumed by devices in routers need to be in proportion to the traffic load because the traffic reduction due to caching is beneficial to reduce the power consumed by these devices. A multicore software router is a good example of such routers because its CPU, memory and NIC devices consume power in proportion to the load on the devices.

Under the assumption that the power consumption of routers is in proportion to their traffic load, we revealed the tradeoff relation of the power consumption of caching as follows: The power increase due to caching comes from the CPU power consumption for the computation of caching, i.e, CS Lookup B_2 and CS Lookup&Insert B_6, at all routers. In contrast, the power reduction due to caching comes from two part; one is the NIC power consumption and the memory device power consumption at upstream routers and the other is the CPU power consumption for the computation of name-based

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

forwarding, i.e., Interest Forwarding B_4 and Receive Data B_5, at downstream routers.

As an example case, we analyze power consumption in commercial networks with NDN routers, which are so well-engineered that the computation of caching is as light as that of name-based forwarding. In this case, caching reduces the power consumed by an NDN network. This is because in such cases the power increase due to caching, which is the cache computation, is smaller than the power reduction due to caching, which is the reduction in name-based forwarding computation. Furthermore, with the well-engineered routers, caching is useful to reduce power consumption when the traffic load in an NDN network is high. This means that the ratio of power reduction due to caching to the power used without caching is large when the traffic load is high. We believe that this is a good feature of caching because reducing peak power in the daytime is an important problem of the society.

Our technique for modeling the power consumed by NDN routers is general enough for PC-based hardware platforms and it is so useful to model power consumption by using just a few parameters. With our power consumption model, we reveal that the power consumed by name-based forwarding and caching accounts for a large portion of the power consumed by an NDN network. Although this may be partly because the reference architecture does not use power hungry specialized devices for routers, it is important to understand how NDN packet forwarding consumes power and to improve the forwarding algorithm more precisely. Our power consumption model can play an important role in this task.

VII. CONCLUSION

This paper developed a power consumption model of a multicore software NDN router focusing on the power consumed by name-based forwarding and caching. We developed the model from a PC hardware platform and an NFD source code assuming that commercial multicore software routers and PC-based routers consume the power similarly. We learned several lessons about power-efficient NDN networks from developing a precise power consumption model. According to our analysis, caching reduces the power consumption of an NDN network if the power consumption of routers is proportional to their traffic load and the computation of caching at the routers is as light as that of name-based forwarding. We believe that our power consumption model of an NDN router will play an important role in developing such techniques.

REFERENCES

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, Jul. 2014.

[2] H. Yuan, T. Song, and P. Crowley, "Scalable NDN forwarding: Concepts, issues and principles," in *IEEE ICCCN*, Aug. 2012, pp. 1–9.

[3] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking (ICN 2011)*, Aug. 2011, pp. 44–49.

[4] W. So, A. Narayanan, and D. Oran, "Named data networking on a router: Fast and DoS-resistant forwarding with hash tables," in *In Proceedings of ACM/IEEE ANCS 2013*, Oct. 2013, pp. 215–226.

[5] G. Rossini, D. Rossi1, M. Garetto, and E. Leonardi, "Multi-terabyte and multi-gbps information centric routers," in *In Proceedings of IEEE INFOCOM*, May 2014, pp. 181–189.

[6] N. Choi, K. Guan, D. C. Kilper, and G. Atkinson, "In-network caching effect on optimal energy consumption in content-centric networking," in *IEEE ICC*, Jun. 2012, pp. 2889–2894.

[7] S. Imai, K. Leibnitz, and M. Murata, "Energy efficient data caching for content dissemination networks," *Journal of High Speed Networks*, vol. 19, no. 3, pp. 215–235, Oct. 2013.

[8] T. Vogelsang, "Understanding the energy consumption of dynamic random access memories," in *Proceedings of IEEE/ACM MICRO*, Dec. 2010, pp. 363–374.

[9] T. Hasegawa, Y. Nakai, K. Ohsugi, J. Takemasa, Y. Koizumi, and I. Psaras, "Empirically modeling how a multicore software icn router and an icn network consume power," in *Proceedings of the 1st ACM ICN*, Sep. 2014.

[10] R. Bolla, R. Bruschi, A. Carrega, F. Davoli, D. Suino, C. Vassilakis, and A. Zafeiropoulos, "Cutting the energy bills of internet service providers and telecoms through power management: An impact analysis," *Computer Networks*, vol. 56, no. 10, pp. 2320–2342, Jul. 2012.

[11] R. Bolla, C. Lombardo, R. Bruschi, and S. Mangialardi, "DROPv2: Energy efficiency through network function virtualization," *IEEE Network*, vol. 28, pp. 26–32, Mar./Apr. 2014.

[12] "NFD - named data networking forwarding daemon," http://named-data.net/doc/NFD/current.

[13] M. Fukushima, A. Tagami, and T. Hasegawa, "Efficient lookup scheme for non-aggregatable name prefixes and its evaluation," *IEICE Trans. on Communications*, vol. E96-B, no. 12, pp. 2953–2963, Dec. 2013.

[14] U. Lee, I. Rimac, D. Kilper, and V. Hilt, "Toward energy-efficient content dissemination," *IEEE Network*, vol. 25, pp. 14–19, Mar. 2011.

[15] U. Lee, I. Rimac, and V. Hilt, "Greening the internet with content-centric networking," in *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, ser. e-Energy '10, 2010, pp. 179–182.

[16] Hewlett-Packard Company, "DDR3 memory technology," http://h20000.www2.hp.com/bc/docs/support/SupportManual/c02126499/c02126499.pdf.

[17] R. Bolla, R. Bruschi, and A. Ranieri, "Performance and power consumption modeling for green COTS software router," in *COMSNETS 2009*, Jan. 2009, pp. 1–8.

[18] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," in *ACM SIGCOMM*, 2013, pp. 147–158.

[19] Intel Corporation, "Intel® 64 and ia-32 architectures optimization reference manual," Sep. 2014, available at https://www-ssl.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html.

[20] "The Linux kernel archives," https://www.kernel.org.

[21] Intel Corporation, "Intel ® Xeon® processor 5600/5500 series platforms for embedded computing," available at http://www.intel.com/content/dam/www/public/us/en/documents/platform-briefs/xeon-5500-5600-platform-brief.pdf.

[22] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *Proceedings of the ACM International Conference on Autonomic Computing*, Jun. 2011, pp. 31–40.

[23] J. Mogul, "Improving energy efficiency for networked applications," Keynote presentation at ACM/IEEE ANCS 2007, Dec. 2007, available at http://www.cse.wustl.edu/ANCS/2007/slides/ANCS2007KeynoteMogul.pdf.

[24] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. L. Braynard, "Networking named content," in *ACM CoNEXT 2009*, Dec. 2009, pp. 1–12.

[25] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *Proceedings of IEEE NOMEN*, Mar. 2012, pp. 310–315.

[26] "Ircache project," http://www.ircache.net/.

[27] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *Proceedings of IEEE INFOCOM*, Apr. 2014, pp. 2040–2048.

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2520278, IEEE Journal on Selected Areas in Communications

Kaito Ohsugi received his Bachelor and Master of of Information Science degrees from Osaka University, Japan, in 2013 and 2015, respectively. He is currently work at KDDI. His research interests include Information Centric Networking.

Junji Takemasa received his Bachelor of Information Science degree from Osaka University, Japan, in 2014. He is currently a master's course student at the Graduate School of Information Science and Technology, Osaka University. His research interests include Information Centric Networking and green networking. He is a member of IEICE, and IPSJ.

Yuki Koizumi received his Master of Information Science and Ph.D. of Information Science degrees from Osaka University, Japan, in 2006 and 2009, respectively. He is currently an Assistant Professor at the Graduate School of Information Science and Technology, Osaka University, Japan. His research interests include Information Centric Networking and mobile networking. He is a member of IEEE, ACM, and IEICE.

Toru Hasegawa is a professor of Graduate school of Information and Science, Osaka University. He received the B.E., the M.E. and Dr. Informatics degrees in information engineering from Kyoto University, Japan, in 1982, 1984 and 2000, respectively. After receiving the master degree, he worked as a research engineer at KDDI R&D labs. (former KDD R&D labs.) for 29 years and moved to Osaka University. His current interests are future Internet, Information Centric Networking, mobile computing and so on. He has published over 100 papers in peer-reviewed journals and international conference proceedings including MobiCom, ICNP, IEEE/ACM Transactions on Networking, Computer Communications. He has served on the program or organization committees of several networking conferences such as ICNP, P2P, ICN, CloudNet, ICC, Globecom etc, and as TPC co-chair of Testcom/Fates 2008, ICNP 2010, P2P 2011 and Global Internet Symposium 2014. He received the Meritorious Award on Radio of ARIB in 2003, the best tutorial paper award in 2014 from IEICE and the best paper award in 2015 from IEICE. He is a fellow of IPSJ and IEICE.

Ioannis Psaras is an EPSRC Fellow at the Electrical and Electronic Engineering Department of UCL. He is interested in resource management techniques for current and future networking architectures with particular focus on routing, caching and congestion control. Before joining UCL in 2010, he held positions at the University of Surrey, and Democritus University of Thrace, Greece, where he also obtained his PhD in 2008. In 2004 he won the Ericsson Award of Excellence in Telecommunications for his diploma dissertation. He has held research intern positions at DoCoMo Eurolabs and Ericsson Eurolabs. More details can be found at: http://www.ee.ucl.ac.uk/~uceeips/

0733-8716 (c) 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.