Empirically Modeling How a Multicore Software ICN Router and an ICN Network Consume Power

Toru Hasegawa
Osaka University
Yamada-oka, Suita-shi,
Osaka, Japan
t-hasegawa@ist.osaka-u.ac.jp

Yuto Nakai
Osaka University
Yamada-oka, Suita-shi,
Osaka, Japan
y-nakai@ist.osaka-u.ac.jp

Kaito Ohsugi
Osaka University
Yamada-oka, Suita-shi,
Osaka, Japan
k-ohsugi@ist.osaka-u.ac.jp

Junji Takemasa
Osaka University
Yamada-oka, Suita-shi,
Osaka, Japan
j-takemasa@ist.osaka-u.ac.jp

Yuki Koizumi
Osaka University
Yamada-oka, Suita-shi,
Osaka, Japan
yuki@ist.osaka-u.ac.jp

Ioannis Psaras
University College London
WC1E 7JE, Torrington Place,
London, UK
i.psaras@ucl.ac.uk

ABSTRACT

ICN (Information Centric Networking) has received much attention due to its built-in functionalities such as caching and mobility-support. One of the important research challenges is to reduce the power consumed by ICN networks because ICN's packet forwarding and packet-level caching are power-hungry. As the first step to achieve power-efficient ICN networks, this paper develops a power consumption model of a multicore software ICN router while taking into account the power consumed by power-hungry computation. This paper makes the following three contributions: First, the model is one of the first realistic models which consider ICN packet forwarding and packet-level caching. Second, the model is represented as a concise set of equations with just a few parameters. Third, we apply the model to estimate power consumed by simple networks.

Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design

General Terms

Measurement, Performance, Design

Keywords

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

1. INTRODUCTION

ICN (Information Centric Networking) [1] has received much attention since it inherently provides attractive functionalities such as mobility support, caching and name-based routing. However, many issues should be resolved so that ICN networks are widely deployed. Among them, time-consuming packet forwarding and packet-level caching raise a couple of issues: forwarding performance [2] and power consumption.

Since name-based packet forwarding and packet-level caching are time-consuming, high performance ICN router implementations are a hot research topic and many studies focus on efficient prefix-matching and caching algorithms [3-6]. On the contrary, the study in [7] designs a protocol such that burdens of name-based longest prefix match (LPM) are mitigated by cooperating ICN routers.

Power reduction is a hot research topic as well, but many studies focus on traffic reduction due to caching, which may reduce power as a byproduct. A well-studied issue is optimizing cache amount allocations in a network [8, 9] assuming that memory devices for caching consume much power at the idle time. For example, Perino *et al*. assume that the DRAM (Dynamic Random Access Memory) devices used for CSs (Content Stores) consume 0.023 W/MBs at the idle time [3]. However, the power consumed by them is being reduced by voltage reduction and manufacturing technology improvement. Voglesng predicts that a decrease in power by bit consumed by DDR (Double-Data-Rate SDRAM) is 1.2 per generation of DDR [10]. The current generation is DDR3, and DDR4 and DDR5 are planned to be develop by 2018. A white paper of some vendor shows that power by the least power-efficient DDR3 device is 0.0000625 W/MBs [11]. Besides it is expected that non-volatile memory devices such as flash memory devices would be used for CSs in the future due to their improvement of reliability and access speed.

This power reduction tendency implies that reducing power consumed by time-consuming packet forwarding becomes important compared with power consumed by memory devices. Thus, in this paper, we address a tradeoff relation between power consumed by packet forwarding/packet level caching and traffic reduction due to caching in terms of power consumed by an ICN network. As the first step, we model how an ICN software router, especially its packet forwarding, consumes power. The targets of this paper are multicore software ICN routers because such routers used in access networks consume more power than routers in backbone networks [12, 13].

Our objectives are two-fold: The first objective is to understand how ICN packet forwarding and packet level caching consume power. This helps us obtain insight on designing power-efficient ICN routers and networks. The second one is to show the necessity of power consumption models of ICN routers so that researchers and engineers develop power-efficient techniques using these models.

The main contributions of this paper are three-fold: First, this is one of the first power consumption models which focus on power consumed by packet forwarding and packet-level caching. This allows us to identify insight on power-efficient parallel ICN packet processing at multicore software routers. Second, the model is represented as a concise set of equations with just a few parameters such as an interest packet rate and a cache hit rate so that the model is easily used in mathematical analysis and simulations for estimating power consumed by an ICN network. Third, we apply the model to estimate the power consumed by a small network. This validates the necessity of the proposed model by estimating the power consumed by ICN packet forwarding.

The rest of paper is organized as follows: Section 2 summarizes the related work. Section 3 proposes the reference architecture. Sections 4 and 5 empirically model how a multicore software router based on NDNx [14, 15] consumes power. Section 4 models the power consumed by the hardware platform and section 5 focuses on power consumed by NDNx packet forwarding. Section 6 applies the model to estimate power consumed by a small network. Section 7 concludes the paper.

2. RELATED WORK

Since name-based packet forwarding and packet-level caching are time-consuming, high-performance ICN router implementations are a hot research topic. Perino *et al*. [3] firstly address this issue and predict future name-based packet forwarding performance. So *et al*. [4, 5] design a high-performance forwarding algorithm and implement it on a commercial router chassis. Focusing on caching, Rossini *et al*. [6] 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 [7]. On the contrary, this paper focuses on power consumed by packet forwarding.

Many studies focus on the caching functionality because traffic reduction would contribute to power reduction in networks. Lee *et al*. [12, 13] investigate how much power is reduced by reducing hop counts to get contents. In their simulations, they use a power consumption model which only considers power consumed by lower layer packet forwarding devices. Choi *et al*. [8] show that the power consumed by memory devices used for caching and for the forwarding processes are not negligible. Imai *et al*. [9] propose a method of deciding capacities of ICN routers' memory devices so that the sum of power consumed by those is minimized.

The power consumption models of these studies consist of the following two analysis techniques: The first technique is analyzing the power consumed by devices. Many power consumption models focus on the power consumed by memory devices because they are power-hungry even at the idle time. On the contrary, this paper focusses on power consumed by packet forwarding, taking into account the memory power reduction tendency [10, 11]. The second technique is analytically calculating cache hit rates of all routers in an ICN network [16-18]. The cache hit rates are used to estimate the amount of packets forwarded by ICN routers and the power consumed by forwarding these packets. This paper also analytically calculates cache hit rates similarly to these studies.

Bolla *et al*. [19] empirically develop a power consumption model of a COTS (Commercial Off-The Shelf) multicore software router. Although this study has been a motivation for us, it only focuses on IP routers and does not consider ICN packet forwarding. Whereas this study formulates power as a 7 dimensional polynomial to model various power optimization techniques, this paper develops the model by avoiding effects from such optimizations.

3. REFERENCE ARCHITECTURE

3.1 Hardware Platform

We choose multicore software routers as our target hardware platforms because we predict that they are going to be used in access networks in the near future. First, full-fledged ICN routers which have all ICN functionalities including caching one would not be used in backbone networks, but only in access ones. This is because caching in backbone networks is not as effective as that in access ones [20]. Second, it is natural that ICN functionalities are not implemented by interface cards, but by service cards like *ISM (Internal Service Module)* cards of some vendor's routers. So *et al*. show that 20 Gbps throughput is feasible on such service cards on commercial routers [4] and the routers are a multicore software router which consists of a service card and multiple interface cards.

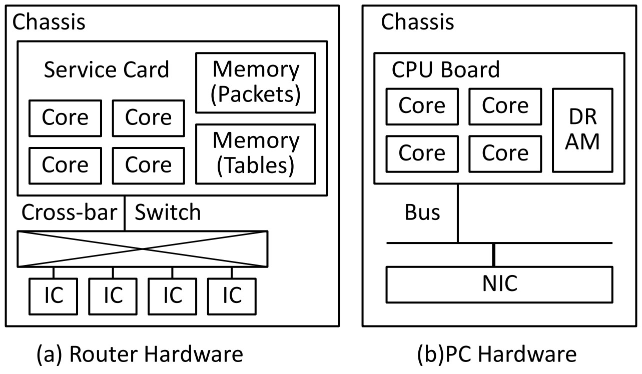

Figure 1. Multicore Software Router and PC Router.

Figure 1 (a) shows a reference multicore software router hardware platform. It consists of a service card and multiple ICs (Interface Cards) which are connected via a cross-bar switch. An ICN protocol is implemented as a program on the service card. Memory devices are used to store tables for name-prefix matching and packets themselves. What kinds of memory devices such as DRAM (DDR3) devices and SSD (Solid State Drive) devices are used is determined depending on requirements to the packet forwarding speed.

The service card is a multicore CPU board with memory devices and its architecture is similar to a PC (Personal Computer) which is shown in Fig 1 (b). We consider that a main difference between a commercial multicore software router and a PC is how devices are connected. For example, they are connected via a cross-bar switch and a bus in a commercial router and a PC, respectively. Thus assuming that these two hardware platforms similarly consume power [4], in this paper, we model power consumed by a PC instead of a commercial multicore software router.

Figure 1(b) shows a reference PC hardware platform. In this paper, we use the minimum configuration which consists of one CPU device 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.

3.2 Software Architecture

We choose NDN/CCN [15] as the ICN protocol because it is widely used and its source code NDNx/CCNx[14] is available. However, since the current NDNx source code is single-threaded, how to extend it to be multi-threaded is an important issue. The two requirements need be considered to do so. First, mutual exclusion among CPU cores should be avoided. Second, none of CPU cores should be lightly loaded so that the number of active CPU cores is minimized. The second requirement comes from the observation that a CPU core fully consumes power even if it is lightly loaded. See section 4.3. In the rest of this section, after describing the hardware platform, we roughly sketch the algorithm because the objective is to show its feasibility.

3.2.1 Data Structures

In order to satisfy the first requirement, we re-design the tables of NDNx which record information used for packet forwarding and packet-level caching. The tables include the NPHT (Name Prefix Hash Table) and CS (Content Store). Each entry of the tables is identified by the name of the content. When incoming NDNx packets are processed by CPU cores in parallel, at least, the tables' entries of the same name should not be simultaneously accessed by multiple CPU cores. Thus we divide the tables to multiple groups of tables as shown in Fig. 2 so that each group of tables have entries of disjoint names with those of the other groups.

An intuitive way of grouping is to assign different name spaces to individual groups of tables similarly to [4]. No CPU core can access any entry which is not assigned to it because each entry of any table is identified by the name of incoming packet. We divide the name space of NDNx to M independent name spaces by hashing a root prefix of name, i.e., the first component of a hierarchical human-readable name, to M hash values. ($M = 4$ in Fig. 2) The names of the same hash value are assigned to the same group. Hashing root prefixes comes from the fact that entries of NPHT that have a parent-child relationship also have a link between them [2].

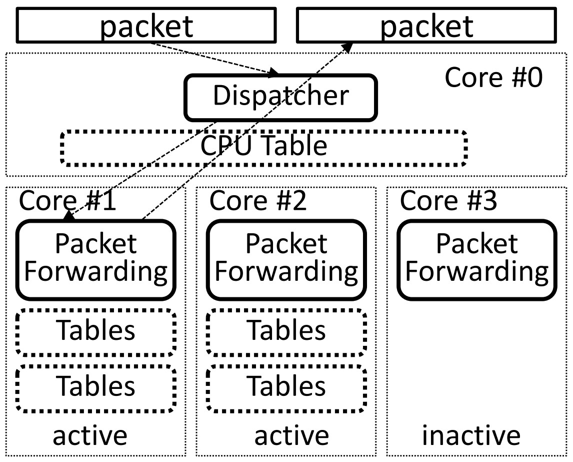

Figure 2. Parallel Processing of NDNx Packets.

3.2.2 Re-assignment Algorithm

In order to satisfy the second requirement, a dispatcher of dispatching incoming NDNx packets to CPU cores is designed as follows. As shown in Fig. 2, the dispatcher which is assigned to some core, e.g., *Core #0*, in Fig. 2, receives an NDNx packet and dispatches it to the CPU core, e.g., *Core #1* in Fig.2. The dispatcher assigns each group of the tables to one CPU core and re-assigns the groups depending on the loads of the corresponding CPU cores. For example, when some CPU cores are lightly loaded, it re-assigns some groups assigned to it to other active CPU cores, so that the CPU core becomes inactive. On the contrary, when some CPU core is heavily loaded, some of their groups of tables are re-assigned to other CPU cores.

In order to do such re-assignments, we introduce a data structure (the *CPU Table* in Fig. 2) which manages pending incoming NDNx packets which are dispatched to individual groups, but are not processed. The dispatcher always monitor the numbers of pending packets and the loads of all CPU cores and re-assigns the groups so that that none of CPU cores is lightly loaded.

We omit the details of the re-assignment procedure. We only address the following three issues to design the algorithm: The first issue is how other processes than NDNx packet forwarding is assigned to CPU cores. We assume that one CPU core, i.e., *Core #0* in the figure, is dedicated to them. Such processes include those of the dispatcher, forwarding IP packets encapsulating NDNx packets and routing protocols.

The second issue is the number of groups. Here, let M and N be the numbers of the groups and the CPU cores, respectively. M is any integer m such that $m \geq N - 1$ because one CPU core is dedicated to the dispatcher. We pick up N as m and thus M is N in the rest of the paper. We assume a simple re-assignment algorithm such that if the incoming packet rates to some CPU cores are heavy, some of the groups are re-assigned to in-active CPU cores. On the contrary, if the rates to some active CPU cores are light, the groups on it are re-assigned to active CPU cores. Since an active CPU core consumes the maximal power as described in section 4.3, the algorithm need not carefully re-assign groups of tables. The third issue is about caching hit rates when a CS is divided to smaller ones due to the above grouping and we address it in section 6.2.

4. POWER CONSUMPTION MODEL

We develop a power consumption model of multicore software NDNx router to satisfy the following requirements. The first requirement is that the model should reflect loads on a hardware platform. It means that the consumed power is a function of the loads. Such loads include a CPU core load, an access to a DRAM (DDR3) device and so forth. The second requirement is that the above loads on the hardware platform should be derived from loads of ICN packet forwarding. Sections 4 and 5 address the first and second requirements, respectively.

4.1 Formulation

We formulate the power consumed by the PC platform of the minimum configuration shown in Fig.1 (b). The power $P_{router}(cores, rate_{IP})$ [W] is defined in equation (1) and it is parameterized by the following two parameters: the number of active CPU cores, i.e., *cores*, and the IP packet forwarding rate, i.e., $rate_{IP}$ [packet/s].

$$P_{router}(cores, rate_{IP}) = P_{cpu}(cores) + P_{mem}(bytes)$$
$$P_{nic}(rate_{IP}) + P_{IDLE} \quad [\mathrm{W}] \qquad (1)$$

- $P_{cpu}(cores)$ [W]: The power consumed by the CPU device. It is the function of the number of active CPU cores $cores$.

- $P_{mem}(bytes)$ [W]: The power consumed by accessing the DDR3 device. It is the function of the number of bytes which are accessed per second.
- $P_{nic}(rate_{IP})$ [W]: The power consumed by the NIC. It is the function of the IP packet forwarding rate $rate_{IP}$.
- P_{IDLE} [W]: The power consumed by the chassis when the router is idle. It includes the power of all devices.

In this section, we empirically measure the above four terms in order to model them. The measurement conditions are as follows: The PC is a PC server with Xeon E3-1220 processor (3.10 GHz*4 cores CPU and DDR3 16 GB memory device). One Intel Ethernet Converged Network Adapter X540-T2 (10 GbE NIC) and a Western Digital 2 T bytes SATA 3.5 inch HDD (Hard Disk Drive) are used. The operating system is Ubuntu 13.10. We use the power meter and the current transformer developed by Omron (ZN-CTX21 and ZN-CTS51-200As). The power is measured in the unit of *Joule/s*, i.e., *Watt*. We set the clock frequency of the CPU device at 1.6 GHz in order to avoid effects from the CPU clock frequency adaptation functions which modern CPU devices have. Each measurement under the same condition is performed twenty times and its measurement duration is ten minute long.

4.2 Power Consumed by Chassis

We measure the power consumed by the PC under the condition that all CPU cores are inactive (idle) and that the NIC is connected to a LAN switch, but any frame is neither sent nor received. The average and distribution are 36.06 [W] and $3.69 \cdot 10^{-6}$ and thus we determine P_{IDLE} to be 36.06 [W]. Besides we measure the average power of the NIC at the idle time $P_{NICIDLE}$ and its two-sided 95% confidence intervals are 13.42 and 13.90 [W].

4.3 Power Consumed by CPU Cores

The power consumed by the CPU device $P_{cpu}(cores)$ is determined by the number of active CPU cores $cores$ as shown by the equation (2). This subsection shows that the power consumed by the CPU device is digitized by the number of active CPU cores.

$$P_{cpu}(cores) = \begin{cases} 5.98\ [\mathrm{W}]\ (cores = 1) \\ 8.20\ [\mathrm{W}]\ (cores = 2) \\ 11.50\ [\mathrm{W}]\ (cores = 3) \\ 14.90\ [\mathrm{W}]\ (cores = 4) \end{cases} \qquad (2)$$

First, to show that power consumed by active CPU cores is digitized by their number, we measure the power consumed by one CPU core at various loads. We run a simple program which repeatedly performs the sequence of computations: arithmetic operations for 12.5 µs and, sleep for a fixed duration, arithmetic operations for 25.5 µs and sleep for a fixed duration. The CPU core is made inactive by using *nanosleep* command of Linux and the minimum sleeping duration is about 55 µs. This sequence emulates NDNx packet forwarding when a cache hit rate is 0, as described in Section 6. The 12.5 µs and 25.5 µs correspond to the duration when the NDNx router forwards one Interest packet to an upstream router and that when it forwards one Data packet to a downstream router, respectively. Thus we can regard the rate of the repetitions [sequence/s] as the input Interest packet rate [packet/s].

Figure 3 shows the measured power at various converted input Interest packet rates [packet/s] and the power that one active CPU core consumes (the dotted line in Fig. 3). From the observation below, we conclude that an active CPU core consumes the maximum power. One CPU core consumes about 90% of the maximum power of one CPU core even if the input Interest packet rate, i.e., 6756.76 [packet/s], is just about 25.7% of the maximum rate which one CPU core provides. Please see the measured power at the rate of 6756.76 [packet/s].

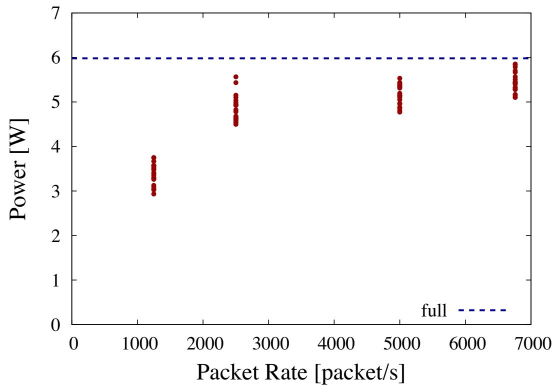

Figure 3. Power Consumed at Various Loads.

Second, we measure the power consumed by 1, 2, 3 and 4 CPU cores by running a program which calculates arithmetic operations. Figure 4 shows the measured power with a scattered graph. The averages for CPU cores are the constants in equation (2).

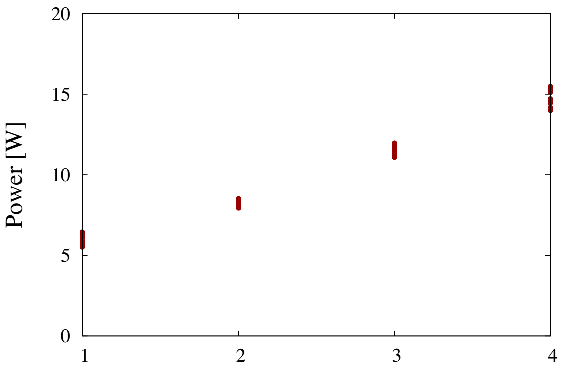

Figure 4. Power Consumed by Multiple CPU Cores.

4.4 Power Consumed by Memory Device

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

$$P_{mem}(bytes) = P_{MEMIDLE} + P_{BYTE} \cdot bytes \quad [\mathrm{W}], \qquad (3)$$

where $P_{MEMIDLE}$ is 1.10 [W] and P_{BYTE} is $0.44 \cdot 10^{-9}$ [Joule/byte]. The objectives of this subsection are to validate that the power consumed by accessing the DDR3 device is proportional to the rate of accessing it [21] and to decide the constants $P_{MEMIDLE}$ and P_{BYTE} in equation (3).

We run the program which repeatedly reads 8 byte data from the array allocated by the *malloc* function. We measure the following values at various sizes of the arrays: 32 MBs, 64 MBs, 256 MBs and 512 MBs.

- The power consumed by the PC hardware platform [W]: We derive the power consumed by accessing the DDR3 device by subtracting the power consumed by one active CPU core (5.98 W) from the measured power.

- The average number of accessed bytes in the DDR device per second (by using the *Intel® Performance Counter Monitor*).

Figure 5 shows the power consumed by the DDR3 device at various access rates [byte/s]. We decide the two constant $P_{MEMIDLE}$ and P_{BYTE} by using least squares approximation. We assume that the power consumed by reading and writing data is the same.

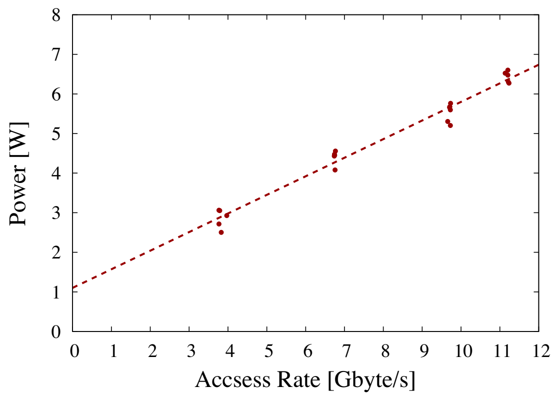

Figure 5. Power Consumed by DDR3 Device.

4.5 Power Consumed by NIC

We formulate the power consumed by the NIC is a function of the IP packet forwarding rate as

$$P_{nic}(rate_{IP}) = P_{PACKET} \cdot rate_{IP} \quad [\mathrm{W}], \qquad (4)$$

where P_{PACKET} is $3.04 \cdot 10^{-6}$ [Joule/packet].

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 are used as a client and server. The client sends UDP packets at various rates by running a simple program which switches between sending a UDP packet and sleeping. We measure the power consumed by the NIC by choosing 1500 bytes as the size of the IP packets.

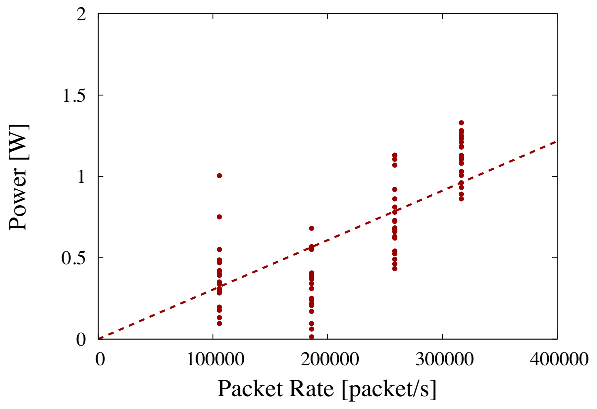

Figure 6. Power Consumed by NIC.

Figure 6 shows the power consumed by the NIC. The power is not exactly proportional to the forwarding rate; however, we assume that is proportional to the forwarding rate $rate_{IP}$. This is because its two-sided 95% confidence intervals are just 2.51 and 2.57 [W] and thus errors between the actual and estimated values would be negligible. We decide the constant P_{PACKET} using least squares approximation.

5. PACKET FORWARDING ANALYSIS

This section addresses how the three parameters *cores*, *bytes* and $rate_{IP}$ of the PC hardware platform's model are defined. These three parameters are defined as functions of the average input Interest packet rate λ_{ICN}^{IN} [packet/s] and the average cache hit rate P_{CS}^{hit} as shown in the equations (5) to (8). This enables equation (1) to be easily used in mathematical analysis and simulations. This is because most studies on caching techniques estimate cache hit rates of all routers under assumed input Interest packet rates to edge routers.

$$cores(\lambda_{ICN}^{IN}, P_{CS}^{hit}) = \lceil \lambda_{ICN}^{IN} \cdot cl_{icn}(P_{CS}^{hit}) / CL_{CORE} \rceil + 1 \quad (5)$$

$$cl_{icn}(P_{CS}^{hit}) = \sum_{f\in F_1} C_f + P_{CS}^{hit} \cdot \sum_{f\in F_2} C_f + (1 - P_{CS}^{hit}) \cdot \sum_{f\in F_3} C_f \quad [\text{cycles}]$$

$$= 17718 + P_{CS}^{hit} \cdot 4917 + (1 - P_{CS}^{hit}) \cdot 31069 \quad (6)$$

$$bytes(\lambda_{ICN}^{IN}, P_{CS}^{hit}) = \lambda_{ICN}^{IN} \cdot Chunk_{SIZE} \cdot R \ [\text{byte/s}] \quad (7)$$

$$rate_{IP}(\lambda_{ICN}^{IN}, P_{CS}^{hit}) = \lambda_{ICN}^{IN} \cdot (2 - 1P_{CS}^{hit}) \quad (8)$$

The constant CL_{CORE} is the maximum CPU clock cycles of one CPU core per second. In this paper, CL_{CORE} is 1.6 G [cycle/s]. The constant R is 11.03 (See section 5.3) and $Chunk_{SIZE}$ is the average chunk size [bytes]. The first term, i.e., $\lceil \lambda_{ICN}^{IN} \cdot cl_{icn}(P_{CS}^{hit}) / CL_{CORE} \rceil$, of equation (5) is the number of cores for the NDNx packet forwarding engine and the second term, i.e. 1, represents the CPU core which carries out all procedures other than NDNx packet forwarding. This corresponds to the core *Core #0* in Fig. 2.

5.1 IP Packet Forwarding Rate

This subsection describes how $rate_{IP}(\lambda_{ICN}^{IN}, P_{CS}^{hit})$ is calculated assuming that Interest and Data packets are encapsulated by UDP/IP packets. First, we explain how Interest and Data packets are forwarded. When the input Interest packet rate and the cache hit rate are λ_{ICN}^{IN} and P_{CS}^{hit}, respectively, Interest and Data packets are received and sent per second in the following way:

- λ_{ICN}^{IN} Interest packets are received from downstream routers,
- P_{CS}^{hit} Data packets are sent back to them
- $\lambda_{ICN}^{IN} \cdot (1 - P_{CS}^{hit})$ Interest packets are sent (forwarded) to upstream routers
- $\lambda_{ICN}^{IN} \cdot (1 - P_{CS}^{hit})$ Data packets are received from the upstream routers
- $\lambda_{ICN}^{IN} \cdot (1 - P_{CS}^{hit})$ Data packets are sent back to the downstream routers

Since each NDNx packet is encapsulated by an IP packet, $rate_{IP}(\lambda_{ICN}^{IN}, P_{CS}^{hit})$is calculated as follows: $\lambda_{ICN}^{IN} \cdot (2 - 1P_{CS}^{hit})$ IP packets are sent and the same number of IP packets are received per second. It means that $\lambda_{ICN}^{IN} \cdot (2 - 1P_{CS}^{hit})$ IP packets are forwarded. Thus $rate_{IP}(\lambda_{ICN}^{IN}, P_{CS}^{hit})$ is $\lambda_{ICN}^{IN} \cdot (2 - 1P_{CS}^{hit})$. We note that in this paper we assume that one IP packet encapsulates one NDNx packet.

5.2 CPU Clock Cycles

The number of active CPU cores $cores(\lambda_{ICN}^{IN}, P_{CS}^{hit})$ is estimated based on the number of CPU clock cycles which the packet forwarding engine uses. It is obtained by ceiling the value which is obtained by dividing $\lambda_{ICN}^{IN} \cdot cl_{icn}(P_{CS}^{hit})$ by the CPU clock cycles of one CPU core per second.

$cl_{icn}(P_{CS}^{hit})$[cycles] is the average number of CPU clock cycles to perform all functions (in the NDNx source code) which are executed after receiving one Interest packet when the cache hit rate is P_{CS}^{hit}. Equation (6) describes how $cl_{icn}(P_{CS}^{hit})$ is calculated. C_f in equation (6) is the average number of CPU clock cycles to execute a block *f*. We call functions which are run sequentially as a block and classify all the functions in the NDNx source code into the following three groups of blocks so that the average number of CPU clock cycles is calculated from a cache hit rate of the router:

- The group of blocks F_1 : The block is always run when an Interest packet is received.
- The group of blocks F_2: The block is run only when an Interest packet hits a Data packet contained in the CS.
- 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.

Thus the individual terms $\sum_{f\in F_1} C_f$, $\sum_{f\in F_2} C_f$ and $\sum_{f\in F_3} C_f$ are the total CPU clock number consumed by executing all the blocks in the group for processing one pair of an Interest and the corresponding Data packets. Since an Interest packet corresponds one-to-one with a Data packet, $\sum_{f\in F_1} C_f + P_{CS}^{hit} \cdot \sum_{f\in F_2} C_f + (1 - P_{CS}^{hit}) \cdot \sum_{f\in F_3} C_f$ calculates the average CPU clock cycle number with the cache hit rate P_{CS}^{hit} when one Interest packet is received. In section 5.4, we classify all the functions into the three groups. In section 5.5, we empirically measure individual bocks' CPU clock cycles.

5.3 Access Rate to DDR3 Device in Bytes

This subsection describes how $bytes(\lambda_{ICN}^{IN}, P_{CS}^{hit})$, i.e., the average number of accessed bytes in the DDR3 per second, is derived. Since it is difficult to precisely calculate how many bytes in the DDR3 device the blocks access at run time, we estimate it in the following way: First, we empirically measure how many bytes in the DDR3 device are accessed by observing a communicating NDNx router. We derive the ratio of the number of accessed (read or written) bytes in the DDR3 to that of bytes of contents which is retrieved. *R* in the equation (7) is this ratio. For example, if *R* is 10, when a 1 GBs of content is retrieved, the NDNx router is assumed to access 10 GBs in the DDR device.

Second, we assume that the above ratio is always the same for any NDNx packets. Here, since $\lambda_{ICN}^{IN} \cdot Chunk_{SIZE}$ is the average byte number of retrieved contents, equation (7) estimates the number of accessed bytes in the DDR3 device per second. We measure the average number of accessed bytes in the DDR3 device per second during the experiments in section 5.3. Each client retrieves between a 1 GBs and 4 GBs of contents from a server. We set the constant *R* as 11.03 in the equation (7) by averaging the measured values.

5.4 NDNx Source Code Analysis

In this subsection, we classify all the functions of the NDNx source code into the three groups of blocks and analyze packet flows among the blocks.

The blocks of group F_1 are the most darkly shaded in Fig. 7. For example, the block *Duplication Check* checks whether the nonce of the received Interest packet is the same as one of the previously received Interest packets or not. If this check is passed, the *PIT (Pending Interest Table)* is looked up to check the Interest packet is already stored in the PIT. If this check is true, the block *Name Prefix HTE Lookup&Insert* creates a NPHT entry and then the block *CS Lookup* checks whether the corresponding Data packet is stored at the CS or not. Otherwise, the blocks which are paled in Fig. 7, e.g., *Insert HTE Lookup&Insert*, *FIB lookup* and *PIT Insert*, are executed. Here, the probability of this check's being true is negligible because the number of outstanding Interest packets of which the corresponding are sent back from the upstream route is nearly 0. Thus the model assumes that the former blocks are always executed and the latter blocks are never executed.

The blocks of group F_2, which are executed when the Interest packet is hit, are medium shaded in Fig. 7. Since the Interest packet is hit at the CS, the block *Interest Consume* finds other pending Interest packets in the PIT related to the name such as Interest packets which are designated the lengths of names and do not exactly match the names but satisfy requirements for matching, and then sends back the Data packets to all the requesting downstream routers.

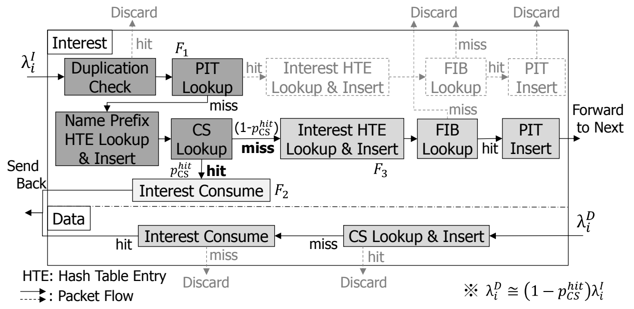

Figure 7. Flow of Blocks of NDNx Software.

The blocks of group F_3, which are executed when the Interest packet is not hit, are lightly shaded in Fig. 7. When the Interest packet does not hit any Data packets in the CS, it is forwarded to the upstream router. The three blocks are executed to forward such an Interest packet. Then the router receives the Data packet corresponding to the forwarded Interest packet, the router executes the two blocks to send back the Data packet after storing it at the CS.

We note that the blocks do not include functions for processing XML-based packets and their parameters. As the authors in [2] show, NDNx packet formats are compliant with XML and processing XML-based packets are 6 to 7 times heavier than the other procedures. We ignore these because future NDNx implementations may change XML-based formats to binary formats so that such packet format processing overhead would be negligible.

5.5 Clock Cycle Measurements

We measure CPU clock cycles of each block as follows: We connect three PCs via a Gigabit Ethernet switch. One PC acts as an NDNx router and two PCs act as a client and a server, respectively. At each server PC, 10 contents are stored and each size is 30 MBs. Four virtual machines are run in a client PC. At each virtual machine, one consumer is run and it requests two of contents at the server PC. Since four virtual machines are running, the client PC sends 8 Interest packets in parallel so that the CPU load of the router is high.

The parameters at the NDNx layer are chosen as follows: The length of content name is 20 characters. The size of CS is 400,000 chunks (about 16 GB) and the size of each content is 30 MBs. 10 contents are stored at each server. These parameters are chosen so that some blocks do not consume extremely large power. For example, the number of components of the name is set to 1. This is because the current implementation of block, *Name Prefix HTE Lookup&Insert* is extremely time-consuming when the component number is large and because the procedures used for name prefix filtering might be deleted in the future.

The NDNx software is run by adding RDTSC (Read Time-Stamp Counter), which reads a time stamp counter, a register incremented by CPU clock cycles in order to measure how many clock cycles each block consumes. Table 1 shows the average CPU clock cycles of individual blocks.

Table 1. The Numbers of Average CPU Clock Cycles

Group	Block f	$Cycles$
F_1	Duplication Check	2068
F_1	PIT Lookup	1029
F_1	Name Prefix HTE Lookup&Insert	4023
F_1	CS Lookup	10598
F_2	Interest Consume (Interest)	4917
F_3	Interest HTE Lookup&Insert	1684
F_3	FIB Lookup	807
F_3	PIT Insert	1232
F_3	CS Lookup&Insert	18176

6. CASE STUDY

This section addresses the following three issues in order to validate the power consumption model. The first issue is how cache hit rates of all routers are affected under the condition where the CS is divided to smaller CSs. The second issue is how our reassignment algorithm contributes to power reduction compared with the fixed dispatching such that groups of tables are assigned to the fixed CPU cores. The third issue is a good example of using our model. We discuss the issues after describing the scenarios used.

6.1 Scenarios

This section shows the conditions of estimations.

- The network topology is a three-level complete binary-tree of NDNx routers. The repository of contents is stored at the server directly connected to the root router.
- The number of contents *G* is 160,000. The size of contents is 10 MBs based on a recent study by Zhou *et al*. which estimates that there are currently $5 \cdot 10^8$ YouTube videos of average size 10 M bytes [22]. Each content is divided to chunks whose size is 1500 byte long.
- The number of CPU cores *N* of all level routers is 4. The number of groups of the tables *M* is 4. The total size of CSs of each router is 16 G bytes and that of each CS is 4 G bytes.
- The popularities of contents are defined in classes. Each class includes 4 contents. The popularities of classes is a Zipf distribution wherein $\alpha = 0.8$ [23].
- One client is connected to the 1st level router. The rate at which each client sends Interest packets is chosen so that the maximum number of CPU cores of the 3rd level router is less than 4. The Interest packet rate is 32,810 [packet/s]. This corresponds to 0.394 Gbps content retrieval.

6.2 Cache Hit Rate Calculation

We calculate the cache hit rates of all routers in the complete-binary tree topology based on the model which Che *et al*. [17] and Fricker *et al*. [18] propose. We assume that a content request process is an independent Poisson process with mean rate λ. We introduce classes to popularities of contents, so that contents of the same popularity distribution are handled by *M* independent caching algorithms based on LRU (Least Recently Used). Thus each class consists of *M* (=4) contents and their popularity is the same. The number of classes *K* is G/M where G is the number of all contents. The popularity of each class q_k is defined in the equation (9).

$$q_k = \frac{1/k^\alpha}{\sum_{n=1}^{K} 1/n^\alpha} \qquad (9)$$

The CS of *C* contents is divided into *M* equal-sized CSs and the size of each CS is $C_m = C/M$ contents. Note that the CS corresponds to the sum of divided CSs of the proposed router in Section 4. Here, we assume that caching algorithms are independently executed under the IRM for CSs and thus that content requests independently arrive at individual CSs. This means that any content in the same class exclusively arrives at the same CS. Here, the number of contents which arrive at each CS is defined as $G_m = G/M = K$. Thus, the mean rate of all content requests λ_m, the popularity of the contents at kth class $q_{m,k}$ and the mean rate of this content $\lambda_{m,k}$ of the divided CS m are defined in equations (10) to (12).

$$\lambda_m = \lambda/M \qquad (10) \qquad q_{m,k} = \frac{1/k^\alpha}{\sum_{n=1}^{G_m} 1/n^\alpha} \qquad (11)$$

$$\lambda_{m,k} = \lambda_m \cdot q_{m,k} \qquad (12)$$

Since we assume the caching algorithms of individual CSs are independently executed, we can derive the cache hit rate of the content at the kth class $p_{m,k}$ by equation (13) and t_{m,C_m} is obtained by solving equations (14). (See the details in [18].)

$$p_{m,k} = 1 - e^{-\lambda_{m,k} \cdot t_{m,C_m}} \quad (13) \qquad C_m = \sum_{k=1}^{G_m} p_{m,k} \quad (14)$$

We can derive the expected value of cache hit rate of the divided CS p_m by equation (15) and finally derive the expected value of the CS p by equation (16).

$$p_m = \sum_{k=1}^{G_m} \frac{\lambda_{m,k}}{\lambda_m} p_{m,k} \quad (15) \qquad p = \sum_{m=1}^{M} \frac{\lambda_m}{\lambda} p_m \quad (16)$$

We assume (i) that the forwarding process of the kth class content of each caching algorithm toward an upstream router is also an independent Poisson process with the mean rate $\Phi_{m,k}$ and (ii) that the arrival process of content requests on an upstream router is the superposition of forwarding processes from downstream routers. $\Phi_{m,k}$ is obtained by solving equation (17).

$$\Phi_{m,k} = \lambda_{m,k} \cdot (1 - p_{m,k}) \qquad (17)$$

From these two assumptions (i) and (ii), we can derive cache hit rates of the 2nd and 3rd level routers by recalculating based on equations (13) to (16).

6.3 Power Reduction Due To Reassignment

We compare the power consumed by routers both with and without our reassignment algorithm of groups of tables as follows.

We calculate the cache hit rates of the 1st , 2nd and 3rd routers based on section 6.2 and they are 0.188, 0.055, 0.043, respectively. We assume that CPU cores of all routers consume the maximum power when the reassignment algorithm is not used *(without re-assignment* in Figures 8 and 9). Figure 8 and 9 show the power at the various Interest packet sending rates up to 32,810 [packet/s].

Figure 8 and 9 show the total power consumed by all routers and the power which is obtained by subtracting the power constantly consumed, i.e., the sum of all routers' P_{IDLE}. In other words, the latter power is the power proportional to loads. The reason why we show Fig. 9 is as follows. The loads of all routers are light due to the small network configuration. The power constantly consumed accounts for a large portion of the total tower. In other words, the power proportional to loads accounts for just a small portion. Thus the difference between the two, i.e., *with re-assignment* and *without re-assignment* in Fig.8 is small even if the power proportional to loads would account for a large portion in actual large-scale networks.

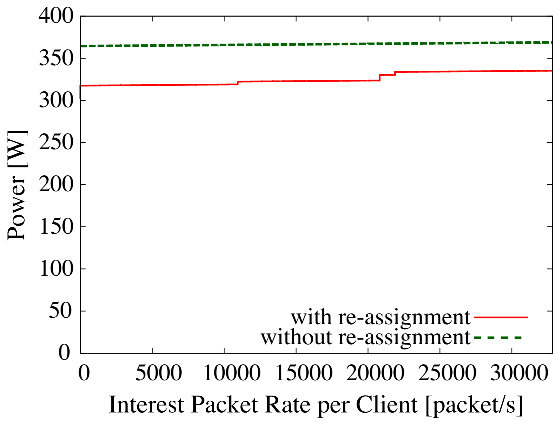

Figure 8. Total Power.

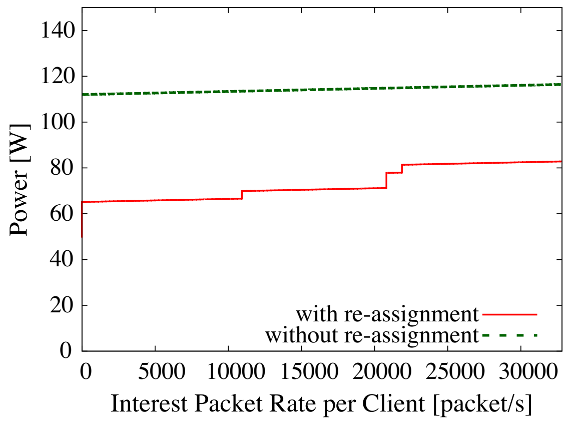

Figure 9. Power Obtained by Subtracting P_{IDLE}.

The observation from the figures is that the reassignment algorithm reduces power consumed by an NDNx network, especially at light loads. However, we note that estimations with the algorithm are the minimum values which would be obtained in ideal conditions.

6.4 Power Reduction Due To Caching

Caching reduces the number of forwarded NDNx packets as the prices for CPU clock cycles for packet level caching, i.e., those of *CS Lookup* and *CS Lookup& Insert blocks* in Table 1. Thus the following question is raised. Whether does caching actually reduce the power consumed by an NDNx network? Thus we try to answer this question by comparing the power consumed by three configurations of networks: all routers, only the 1st level routers and none of routers provide the cache functionality. The three configurations called as *cnfg-all*, *cnfg-1st* and *cnfg-none*, respectively.

Table 2 show the power consumed in the three configurations. The second and third rows show the total power and the power proportional to loads. The difference in the row *(b)* somewhat remarkable compared with that in the row *(a)*. The power of *cnfg-all* is the largest among three configurations. At least, in this small network, the packet number reduction due to caching does not compensate for the prices for CPU clock cycles for packet level caching.

This observation is somewhat contradictory to the intuition that caching would reduce power consumptions. What we want to say from the observation is not that caching is not effective to reduce power, but that the precise power consumption model estimating the power consumed by packet forwarding is inevitable both to estimate power consumed by an NDNx network and to analyze tradeoffs between the reduction of forwarded packets and the increase of power due to packet level caching.

Table 2. Power Consumed by NDNx Network [W]

	cnfg-all	cnfg-1st	cnfg-none
(a)Total power	335.2	325.3	326.0
(b)Power proportional to load	82.9	73.0	73.7

6.5 Lessons Learned

The power consumption model which we develop and the case studies are pre-mature, but we obtain important lessons to achieve power-efficient ICN networks.

- The power consumed by ICN packet forwarding and packet level caching accounts for a large portion of the power consumed by an ICN network. Although this may be partly because the current ICN routers' designs are not mature, it is important to understand more precisely how ICN packet forwarding consumes power and to improve the forwarding algorithm. Our power consumption model focussing on packet forwarding would play an important role to do so.
- The proposed algorithm of re-assigning loads, e.g., the groups of tables, among CPU cores is such an example of algorithm. The algorithm is designed so that the number of active CPU cores is minimized. Multicore software ICN routers should be designed to consider efficiency in both forwarding performances and power consumed by packet forwarding.
- Caching itself does not reduce power due to the prices for CPU clock cycles for packet level caching in some environments. This shows that reducing the power is not trivial and it is an important research topic.
- Switching-off devices when their load is light is useful to reduce power. The proposed re-assignment algorithm is such an example. Another promising candidate is to switch-off NICs because the power at the idle time is high. For example, the power of the NIC in this paper is between 13.42 and 13.90 W and it is about three times larger than that of one CPU core. This observation implies that caching would play an important role to achieve traffic engineering techniques for switching-off redundant links [24].

- Although the model is hardware platform dependent, we consider that the models on the other hardware platforms are easily developed. First, our modelling technique is concise and it specifies the power as functions with a few parameters. Second, most assumptions on power consumptions are confirmed in section 4 and we believe that they would be true on other platforms. An example assumption is that power consumed by the DDR device is proportional to the access rate [byte/s].
- In addition, although the model focuses on a specific PC which is implemented in terms of current technology, we believe that the modelling method which the paper proposes is applicable to model multi-core software routers in the future due to the following reasons: First, the hardware platforms of multicore software routers and PCs of current technology are similar as described in section 3.1. Second, since most technologies of devices including memory devices are mature, it is expected that their performances and consumed energy are gradually improved.

Finally, we note that the current model does not precisely estimate power consumed by longest-prefix matching when the number of entries in the NPHT is large. This modelling is necessary to model backbone routers and thus we are on the way to extend the model so as to provide such modeling.

7. CONCLUSION

This paper develops a power consumption model of a multicore software ICN router focusing on power consumed by packet forwarding and packet-level caching. We develop the model from a PC hardware platform and the NDNx/CCNx source code assuming that commercial multicore software routers and PC-based routers similarly consume power. We obtain several lessons from developing the precise power consumption model. We believe that modeling power consumptions is as an important research topic in networking communities as in hardware/system communities wherein power consumption models of memory devices [10] and server systems [21] are developed.

8. ACKNOWLEDGMENTS

The research leading to these results has partially received funding from the EU-JAPAN initiative by the EC Seventh Framework Programme (FP7/2007-2013) Grant Agreement No. 608518 and NICT under Contract 167 (the GreenICN project).

9. REFERENCES

[1] Dannewitz, C., Imbrenda, C., Kutscher, D. and B. Ohlman. Survey of Information-Centric Networking. IEEE Communication magazine, Vol. 50, Issue 7, pp.26-36, July 2012.

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

[3] Perino, D. and Varvello, M. A Reality Check for Content Centric Networking. In Proceedings of ACM ICN'11, pp. 44-49, Aug. 2011.

[4] So, W., Narayanan, A., Oran, D. and Stapp, M. Named Data Networking on a Router: Forwarding at 20Gbps and Beyond. In Proceedings of ACM SIGGOMM 2013, pp. 495-496, Aug. 2013.

[5] So, W., Narayanan, A., Oran, D. Named Data Networking on a Router: Fast and DoS-resistant Forwarding with Hash Tables. In Proceedings of ACM/IEEE ANCS '13 pp.215-226, Oct. 2013.

[6] Rossini, G., Rossi1, D., Garetto, M. and Leonardi, E. Multi-Terabyte and Multi-Gbps Information Centric Routers. In Proceeding of IEEE Infocom 2014, pp.181-189, May 2014.

[7] Fukushima, M., Tagami, A. and Hasegawa, T. 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.

[8] Choi, N., Guan, K., Kilper, D. and Atkinson G. In-network caching effect on optimal energy consumption in content-centric networking. In Proceedings of 2012 IEEE ICC, pp. 2889–2894, June 2012.

[9] Imai, S., Leibnitz, K. and Murata M. Energy efficient data caching for content dissemination networks. Journal of High Speed Networks, vol. 19, pp. 215–235, Oct. 2013.

[10] Vogelsang, T. Understanding the Energy Consumption of Dynamic Random Access Memories. In Proceedings of 43rd IEEE/ACM MICRO, pp. 363-374, Dec. 2010.

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

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

[13] Lee, U., Rimac, I., and Hilt, V. Greening the internet with content-centric networking. In Proceedings of the first International Conference on Energy-Efficient Computing and Networking, pp. 179–182, Apr. 2010.

[14] http://named-data.net/codebase/platform/

[15] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N. and Braynard, R. Networking named content. In Proceedings of ACM CoNEXT 2009, pp. 1–12, Dec. 2009.

[16] Psaras, I., Clegg, R., Landa, R., Chai, W. and Pavlou, G. Modelling and Evaluation of CCN-caching Trees. In Proceedings of Networking'11, pp.78-91, May 2011.

[17] Che, H., Tung, Y. and Wang, Z. Hierarchical Web caching systems: Modeling, design and experimental results. IEEE J. Selected Areas in Communications, vol. 20, pp. 1305–1314, Sept. 2002.

[18] Fricker, C., Robert, P. and Roberts, J. A Versatile and Accurate Approximation for LRU Cache Performance. In Proceedings of ITC'12, pp.1-8, Sept. 2012.

[19] Bolla, R., Bruschi, R. and Ranieri. Performance and Power Consumption Modeling for Green COTS Software Router. In Proceedings of COMSNETS 2009, pp. 1-8, Jan. 2009.

[20] Fayazbakhsh, S., Lin, Y., Tootoonchian, A., Ghodsi, A., Koponen, T., Maggs, B., Ng, K., Sekar, V. and Shenker, S. Less pain, most of the gain: incrementally deployable ICN. In Proceedings of ACM SIGCOMM 2013, pp. 147-158, August 2013.

[21] Kim, M., Ju, Y., Chae, J. and Park, M. A simple Model for Estimating Power Consumption of a Multicore System Server. International Journal of Multimedia and Ubiquitous Computing, Vol.9, No.2, pp. 153-160, 2014.

[22] Zhou, J., Li, Y., Adhikari, K. and Zhang, Z-L. Counting YouTube Videos via Random Prefix Sampling. In Proceedings of ACM IMC'11, pp.371-380, Nov. 2011.

[23] Fricker, C. Robert, P., Roberts, J. and Sbihi N. Impact of Traffic Mix on Caching Performance in a Content-Centric Network. In Proceedings of IEEE NOMEN 2012, pp. 310-315, March 2012.

[24] Xu, L. and Yagyu, T. Multiple-tree based Online Traffic Engineering for Energy Efficient Content Centric Networking. IEICE Technical Report, IA2013-78 , vol.113, no.424, pp.61-66, Jan. 2014.