Received: October 12, 2022. Revised: November 14, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Closer Towards Named Data Networking Implementation

Tody Ariefianto Wibowo[1]* **Nana Rachmana Syambas**[1] **Hendrawan**[1]
Leanna Vidya Yovita[2] **Ade Aditya Ramadha**[2]

[1]*School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Indonesia*
[2]*School of Electrical Engineering, Telkom University, Indonesia*
* Corresponding author's Email: tody.wibowo@students.itb.ac.id

Abstract: Due to named data network (NDN) 's advantages as a content-based architecture, it is a strong candidate for future internet architecture. Extensive research has been done on NDN implementation in the real world. However, the method widely used has advantages and disadvantages in reaching the goal. Simulation gives simplicity but is unlikely to provide reality in node implementation. The testbed method is close to real-world performance but needs higher effort. To solve the problems, building an environment with nature closer to the implementation world is required. The method also gives ease of use and observation, like simulation/emulation. Due to these needs, we proposed a framework that combines the advantages of emulation and a testbed that occupies a UNetLab-based virtualization environment for extensive observations for the NDN researcher. We build a test case in the PNETLab and Mini-NDN for gives a perspective on how our proposed method works. PNETLab framework evaluates NDN network's nodes performance, giving 7,59% in RTT delay, and 0,89% in nCHR deference result to Mini-NDN, and provides intensive hardware performance parameters result which is very important for NDN implementation.

Keywords: Named data network, Implementation, Virtualization, Performance.

1. Introduction

Named data networking (NDN) was initially proposed by Van Jacobson [1, 2] for facing the critical problem due to the increased Internet usage for end-to-end connection. NDN networks emphasize receiving and sending content or data instead of looking for content/data itself. It has a big difference from IP. In NDN, consumers request the data using the names. The advantage is that every node in the network can respond to this request. While in an IP network, the request from the consumer is addressed to the specific node. It can cause an unnecessary burden on the network when many consumers request the same data. All these requests will be directed to a server with the same address. Due to the advantages of named data network as a content-based architecture, NDN is a strong candidate for future internet architecture.

The general concept of communication mechanism that uses the name of data is called information-centric networking (ICN). To realize the vision of ICN, various architectures have been proposed including DONA [3], PURSUIT [4], NetInf [5], COMET [6], CONET [7], CONVERGENCE [8], MobilityFirst [9], GreenICN [10], CCN [1], and NDN. NDN research is one of the most active ICN research projects for all of these studies. This is because many academic institutions, telecommunications companies, and non-profit research centers are involved in the NDN research consortium, led by the University of California Los Angeles as consortium administrator. Moreover, NDN has several large and prestigious meetings that take place regularly and are used to strengthen and disseminate research on NDN, for example, the ACM ICN conference, MILCOM conference, NDN hackathon, and NDN community meetings.

The US government NSF initially funded NDN as their proposed architecture for shaping the future internet architecture (FIA) in 2010. Nowadays, many universities around the globe take part, whereas UCLA university is the coordinator. Previous considerable studies have carried out initiatives to implement NDN schemes on data communication networks. Still, there have been no structured steps to implement NDN on commodity hardware in the real world.

As illustrated in Fig. 1, many initiatives or research attempts to implement NDN in the real world. In general, the stages of the research carried out were simulation, emulation, and testbed. In the earlier step, simulations in NDN research usually use NDNSim [11]. NDNSim was developed from the NS-3 simulation framework and utilized widely by the researcher. NDN IoT-based research [12, 13] uses NDNSim simulation to verify their proposed algorithm. NDNSim benefits from the NS-3 framework by leveraging the features of existing utilities and helpers.

Nevertheless, every NDN forwarding daemon (NFD) and ndn-cxx release must be manually integrated with NDNSim. NFD works as a network forwarder, and ndn-cxx is a C++ library for implementing NDN core. They are an essential part of creating NDN nodes. Even more, some elements in NDNSim sometimes show an anomaly on the run because some simplifications were made in the simulation. Therefore, we need to ensure that existing and newly added unit tests are running consistently

Emulation is the next step in NDN research. In [14] M. S. Budiana et al. using Mini-NDN to investigate impact of contentstore to cache replacement scheme. Xian Guo et al. [15] proposed NOLSR scheme for NDN-MANET routing protocol using a Mini-NDN emulator. Emulation intends to approach hardware and operating systems mechanisms closer to real-world processes. The majority of NDN researchers use Mini-NDN to emulate the NDN process. Mini-NDN is a lightweight networking emulation tool built upon Mininet. It enables testing, experimentation, and NDN research to reach scalability. NDN libraries, NFD, NLSR, and tools released by the NDN project to emulate an NDN network. Mini-NDN runs on docker as a container-level virtual machine (VM) environment. Therefore Mini-NDN can emulate complete NDN nodes and networks on a single system. Each packet delivery and process in Mini-NDN could be analyzed using a packet capture tool (Wireshark or Tcpdump) and a process logging file (NFD and NLSR). However, Mini-NDN lacks isolating process, and CPU/RAM resource allocation for each node.

The closest step to real-world implementation is the testbed. In [16], Divya Saxena and Vaskar Raychoudhury simulated and built a testbed of NDN-based Smart Healthcare using IoT devices. L. Gameiro, C. Senna, and M. Luís in [17] implement Intelligent Transportation System based on NDN using Raspbery Pi and Nvidia Jetson. In the testbed, usually, the researcher uses a dedicated resource to implement the NDN node on a lab scale [18]. A dedicated scale means the researcher must be allocated real resources for running NDN nodes. With dedicated resource allocation, every process could be monitored: CPU utilization, RAM usage even energy consumption per node. As a consequence, this method has become less cost-effective than others.

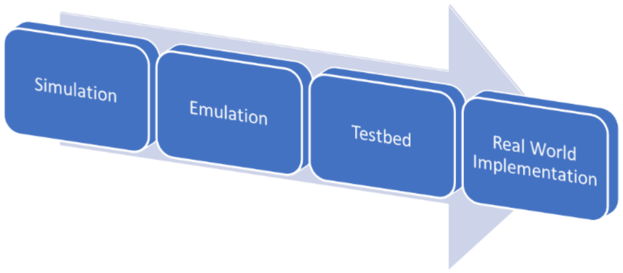


Figure. 1 NDN real-world implementation steps

To solve the problems, building the environment with nature closer to the implementation world has ease of use, and observation is necessary. Due to these needs, we proposed a framework combining emulation and testbed advantages. This combination framework gathers NDN research goals to be as close as possible to real-world implementation. We proposed a UNetlab-based emulation process using a Qemu-based VM to gather easy implementation, extensive monitoring, and cost-effectiveness. This paper shows how this framework can monitor QoS parameters, CPU utilization, and RAM usage. The CPU and RAM usage are impossible to check and analyze if we only use the NDNSim or even Mini-NDN. We believe that this framework could bring researchers the first step closer to implementing NDN networks and technologies.

The rest of the paper is organized as follows. We present the fundamental of NDN virtualization in section 2. Section 3 presents the proposed framework's system model and all the components involved. In section 4 we present the proposed framework and experimental setup, results, and discussion. Finally, we conclude the paper in section 5 and give a direction for future works in section 6.

2. NDN virtualization

Virtualization is essential for modeling the NDN nodes because we can enhance distributed systems' flexibility, process isolation, and fault tolerance.

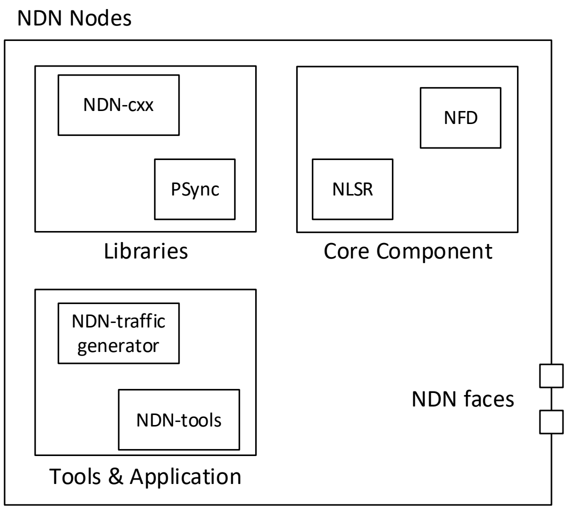


Figure. 2 NDN nodes component

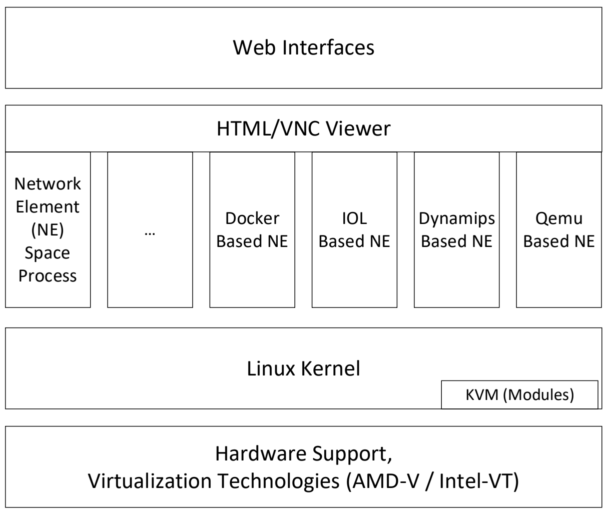


Figure. 3 UNetlab structure

Process isolation is vital because each node runs independently with dedicated hardware resources in the real environment. Without the isolation, it won't be easy to monitor the performance of NDN nodes.

Previously, IBM initialized virtualization technology in 1973, and through time virtualization became very popular and extensively used. Nowadays level/type and application of virtualization have become more varied. Starting from the initial full virtualization and para-virtualization evolve to scalable operating system-level virtualization.

Implementation of virtualization, for example, is used to operate several virtual machines (VM) on one physical machine or server. This mechanism will improve server resource usage in terms of CPU, RAM, and energy. One dedicated server usually uses only a few of the available hardware resources. Using VM also increases the company's flexibility for running the server operation. Adding a resource, hot backing up, migrating, and other server operations are straightforward. These benefits, in the end, will give the company better operations reliability.

Virtualization is also adopted in the data communication world. There is a term called virtualization network function (VNF). VNF, virtualize network and other functions, such as network security, out of dedicated hardware devices. Both network service providers and enterprises have widely used this VNF method in virtualizing routers, switches, firewalls, load balancers, DNS servers, and caching mechanisms.

In this paper, we proposed a framework based on the virtualization mechanism in NDN research as an added step for real-world implementation. In this step, the researcher virtualizes the NDN function before going to the testbed step. In the NDN virtualization step, they virtualize NDN functions in software, so a new algorithm or function proposed by the researcher can be deployed as VMs or containers. With this steppingstone, it will be more quickly and efficiently to do NDN research. Furthermore, in NDN virtualization, the researcher can monitor the system's performance, which is set as close as the hardware resource of the testbed candidate system.

2.1 Named data networking nodes

NDN nodes are built upon a core library/code based called NDN-cxx. NDN-cxx collaborates with core components (NFD & NLSR), NDN tools, and applications (NDN traffic generator), as depicted in Fig. 2. NDN-cxx is used as an NDN code library firstly built and used to write various NDN applications. The core component function is NDN components that are mandatory for running NDN node functions, such as routing, caching, and forwarding. While the NDN tools and application is used as an extension for the experiment. NDN node faces are similar to interfaces in routers connected to other NDN nodes, consumers, or producers.

There are two types of NDN packets, interest and data. The consumer uses interest packets to request content, while the data packet is sent to the consumer in response to an interest packet. Whereas the functions differ, interest and data packets use the name as the content identifier. Each node in the NDN network also has a name, so the routing mechanism uses a name.

2.2 UNetlab

UNetlab is a free multi-vendor and multi-user emulator platform for creating and modeling various network functions and topologies. Some say it emulates, but the method of emulating the network

Received: October 12, 2022. Revised: November 14, 2022.

function has a higher level than in Mini-NDN. UNetlab can virtualize routers, switches, firewalls, customers, and producers. The emulator has vast support for many types of equipment from various vendors, such as Cisco, Citrix, PaloAlto, Juniper, VMWare, Windows Server, Aruba, and Fortinet. Because NDN has not been standardized, it is not listed as a supported function. Even so, we can build the NDN function upon supported operating systems functions such as Ubuntu or other UNIX-based operating systems.

EVE-NG [19] and PNETLab [20] are part of the UNetlab emulator development. Prioritizing the ease of use, friendly user interface, and providing virtualization options up to the container level these why it has become our preference. EVE-NG and PNETlab have almost the same user interface and features. However, with several considerations, such as the number of free licenses, examples of cases that have been carried out, and social media support, in this study, we decided to use PNETLab.

NDN nodes are created over PNETLab-based Qemu virtualization and installed on a small Bodhi Linux 32bit OS. We used Bodhi Linux because of its non-demanding hardware resources. Bodhi Linux's minimum requirements are a 500MHz processor, 512MB of RAM, and 5GB of drive space. Since Bodhi Linux is built based on Ubuntu LTS, we have not found many difficulties installing NDN components to this OS.

2.3 UNetlab network element & QEMU

UNetlab support four kinds of virtualization environments for the Network Element space process. They are Docker, IOL, Dynamips, and Qemu, as depicted in Fig. 3. All of them are running upon hardware support (AMD-V or Intel-VT) depending on hardware specifications and Linux kernel KVM modules. Every compatible hardware has specific requirements to emulate. Furthermore, the choice of virtualization environment will also affect the performance of the emulated hardware.

Docker [21] is a container level of the virtualization environment. Docker is well-known and widely used because of its flexibility and adaptive resource management. In UNetlab, supported devices/features are minimal. A docker for Wireshark, chrome, Ubuntu, ansible, and Cisco NSO is most used for testing, capturing, and device orchestration.

Internetwork operating system over Linux (IOL) is an emulation dedicated to Cisco IOS systems devices. IOL can support Cisco routers, switches r, firewalls, SD-WAN, and servers running on Linux operating system.

Dynamips [22], like IOL, are also built to emulate cisco devices. Dynamips is widely used as experimental telecommunications testing benchmark and for cisco exam preparation. Powered by the GNS3 community, Dynamips git is actively developing the system. Initially, Dynamips only supported Cisco 7200 router then; currently, it can emulate other series of Cisco routers, such as the Cisco 3600, 3700, and 2600 routers family.

QEMU is an abbreviation for Quick Emulator, a generic and open-source machine emulator and virtualizer. There are two types of QEMU emulation levels. The first type is System emulation. In this mode, QEMU emulates a virtual mode of an entire machine. It includes CPU, memory, and emulated device running a guest OS. The second type is User mode emulation. QEMU can invoke a Linux executable compiled for a different device architecture in this mode. QEMU can leverage the host system resources to launch processes compiled to a different architecture in this mode. Because of these complicated mechanisms, some features may have compatibility issues. QEMU commands for user mode emulation are named qemu-target_architecture, e.g., qemu-x86_64 for emulating Intel 64-bit CPUs.

To virtualize NDN nodes in the UnetLab environment, we are using Qemu-based virtualization. The selection of Qemu is due to NDN implementation on the Linux-based OS (Bodhi and because of the flexibility of Qemu virtualization environment support.

3. System model

In this research, we build some study cases to show how this proposed system can help NDN researchers for an intensive experiment. In our study, this NDN system is deployed to analyze the performance of the proactive routing protocol in the NDN-based Indonesia digital network (IDN).

3.1 IDN

IDN plays an essential role in the Indonesian development strategy. Data communication infrastructure is believed to be an essential step to strengthening Indonesia's position in facing the challenges of global change. To realize this vision, the government and stakeholders build the Indonesia digital network (IDN), which focuses on developing and providing internet connectivity for all regions of Indonesia. The digital telecommunication

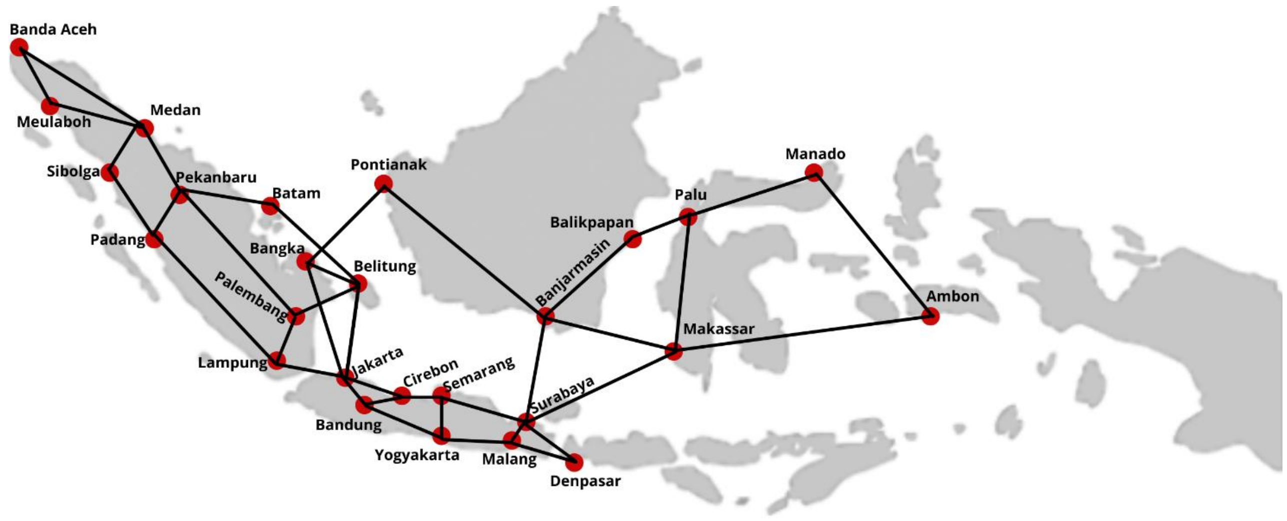


Figure. 4 IDN model

infrastructure includes fiber optic and terrestrial cables on the land and submarine cables at sea, as seen in Fig. 4.

To see how intensive PNETLab (PL) can be used for emulating NDN based network, we create IDN-based NDN in the PNETLab virtual lab and then compare the scenario with widely used Mini-NDN (MN) based NDN emulation. Many NDN researcher adopts Mini-NDN emulation as the primary emulation tool for NDN. The IDN model created is the backbone part that stretches from Banda Aceh to Ambon in the east of Indonesia. This model involves twenty-six nodes, consisting of sixteen main cities and ten transits, where NDN nodes are placed, as seen in Fig. 4. In the scenario, Jakarta plays the role of a producer, and (Medan, Semarang, Pontianak, Makassar, Denpasar, Yogyakarta, Padang, and Manado) as a consumer.

3.2 Content request

In this system, consumers generate content requests based on Zipf-Mandelbrot [23]. We model the traffic from consumers' content requests using the Zipf-Mandelbrot distribution. Contents with top-ranking popularity are more likely to be requested by consumers than non-popular content. The consumer's probability request of content *i* is expressed by Eq. (1):

$$p(i) = \frac{(i+q)^{-\alpha}}{\sum_{i=1}^{N}(i+q)^{-\alpha}} \tag{1}$$

Where:

p (i) = probability request of content i,
q = flattened factor,
α = exponential factor,
N = number of files or content.

3.3 Network cache hit ratio (nCHR)

In this research, the term 'network cache hit' denotes the number of requests for content that intermediate NDN routers can respond. Intermediate NDN router means NDN nodes between consumer and producer. So, one request is said to be a 'hit' when the interest packet can be satisfied by the content cache by the intermediate node so that the interest packet is not forwarded to the producer. Therefore, the NDN intermediate router's ability to satisfy the interest can improve the QoS without burdening the producer's resources. So that the cache hit ratio is the number of content interest that is successfully hit (network cache hit) compared to the amount of total interest (hit and miss). The network cache hit ratio (*nCHR*) [24] for the scenario is obtained using the following formula:

$$nCHR = \frac{nH}{nH+nM} \tag{2}$$

Where:

nH = number of hit interests at intermediate nodes,
nM = number of missed interests at intermediate nodes.

3.4 CPU and RAM utilization

CPU utilization is a parameter of average CPU usage over the observation period. The term CPU utilization percent is applied for a time period portion that a CPU component is working and running the processing, divided by the total amount of observation time. The result is multiplied by 100 to obtain a percentage, denoted in Eq. (3).

There are at least 4 (four) groups of processes in NDN default nodes, routing process, caching process, NDN forwarding process, and others background processes; see Eqs. (3) and (4). For NDN CPU Utilization measurement, elaborate in the next section, focused on NDN consumer and producer CPU utilization percentages. In NDN consumers, there is a traffic-generating process, so Eq. (4) evolves to Eq. (5). Meanwhile, there is no traffic-generating process in the NDN producer, and not every interest packet reaches the producer. Because of the cache hit mechanism, Eq. (6), so that Eq. (4) evolve into Eq. (7)

$$U = \frac{\sum_{i=1}^{p} tCPUwork_i}{T} \times 100\% \tag{3}$$

$$U_{NDN} = \frac{tRP+tCP+tFP+tBP}{T} \times 100\% \tag{4}$$

$$U_{cons} = \frac{tRP+tCP+tFP+tBP+tTG}{T} \times 100\% \tag{5}$$

Where:
RP = NDN Routing Process
CP = NDN Caching Process
FP = NDN Forwarding Process
BP = Others Background Process
TG = Traffic Generating Process

$$P(M) = 1 - P(H) \tag{6}$$

$$U_{prod} = \frac{tRP+tCP+\prod_{i=1}^{n} P_i(M)tFP+tBP}{T} \tag{7}$$

Where:
$\prod_{i=1}^{n} P_i(M)$=*Network cache missed*
n=Nodes along path

Here is the meaning of the average CPU usage rate of 75% during the observation time. It means the CPU occupies 75% of the time executing the process along with observation time. When CPU utilization reaches 100% usage rate, so the CPU works all the time of observation. In this case, the process can overload the CPU capacity. When process overload happens, then it can be seen by increasing RAM usage. We built our code in [24] for CPU, RAM, and VRAM measurement every 1 ms, saved in a CSV file.

3.5 Throughput

Throughput is defined as the average number of data bits transmitted in a time unit. NDN data packet transmission depicts in Fig. 5. Data communication

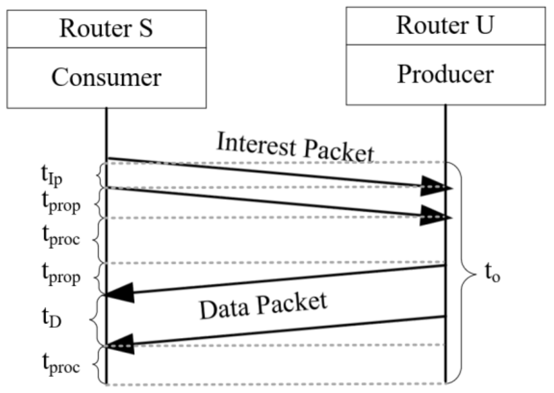


Figure. 5 NDN data transmission [25]

starts when the consumer sends an interest packet for the desired content. Involve several mechanisms and can be said to be over until the consumer receives all data packets from the producer. The total time needed for that process is in Eq. (8). The whole desired data can have a large size file denoted as D Byte, such as multimedia, picture, or music. Thus, NDN divided the D file into several c chunk data packets and transmitted them i times. With the condition of the non-blocking channel and NDN nodes, the resource is enough for data packet processing so that T throughput can be proportional to PS packet size, as elaborate in Eqs. (9)-(13).

$$t_o = 2tprop + 2tproc + t_{Ip} + t_D \tag{8}$$

$$T = \frac{D}{\sum_{i=1}^{c} t_i} \tag{9}$$

$$T = \frac{D}{\sum_{i=1}^{c}(2tprop+2tproc+t_{Ip}+t_D)_i} \tag{10}$$

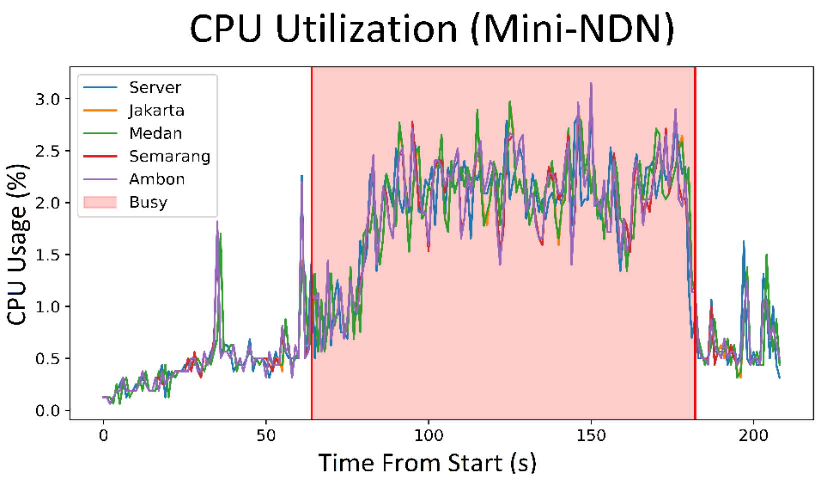
$$\sum_{i=1}^{c} t_i \propto ct_i = \frac{D}{PS} t_i \tag{11}$$

$$T \propto \frac{D}{D/PS(2tprop+2tproc+t_{Ip}+t_D)} \tag{12}$$

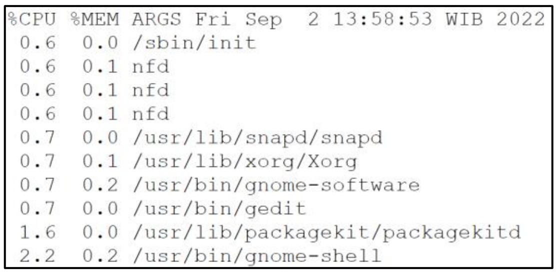
$$T \propto \frac{PS}{(2tprop+2tproc+t_{Ip}+t_D)} \tag{13}$$

Where:

t_o	= *Transmission time (s),*
$tprop$	= *Propagation time (s),*
$tproc$	= *Packet processing time (s)*
t_{Ip}	= *Interest packet transmission time (s)*
t_D	= *Data packet transmission time (s)*
D	= *Data packet size (B)*
PS	= *Packet Size (B)*



(a)



```
%CPU %MEM ARGS Fri Sep  2 13:58:53 WIB 2022
 0.6  0.0 /sbin/init
 0.6  0.1 nfd
 0.6  0.1 nfd
 0.6  0.1 nfd
 0.7  0.0 /usr/lib/snapd/snapd
 0.7  0.1 /usr/lib/xorg/Xorg
 0.7  0.2 /usr/bin/gnome-software
 0.7  0.0 /usr/bin/gedit
 1.6  0.0 /usr/lib/packagekit/packagekitd
 2.2  0.2 /usr/bin/gnome-shell
```

(b)

Figure. 6 (a) NDN nodes & host server CPU Utilization on Mini-NDN; (b) Process CPU & memory usage capture for NDN nodes and host server

Table 1. Study case parameter

Parameter	**Value**
Number of NDN nodes	26
Number of producers	1 (Jakarta)
NDN nodes CPU core	1
NDN nodes RAM	2048 MB
Number of consumers	2, 5, 8 Consumers
Packet size	1024, 4096, 8192
Interest rate	100 int/s
Exp Factor	0,8
Flattened factor	3
Cache size	30 packets
Link bandwidth	10, 100 Mbps
Scenario	26 NDN node virtualization

3.6 Other parameters

Detail of the PL study case parameters shown in Table 1.

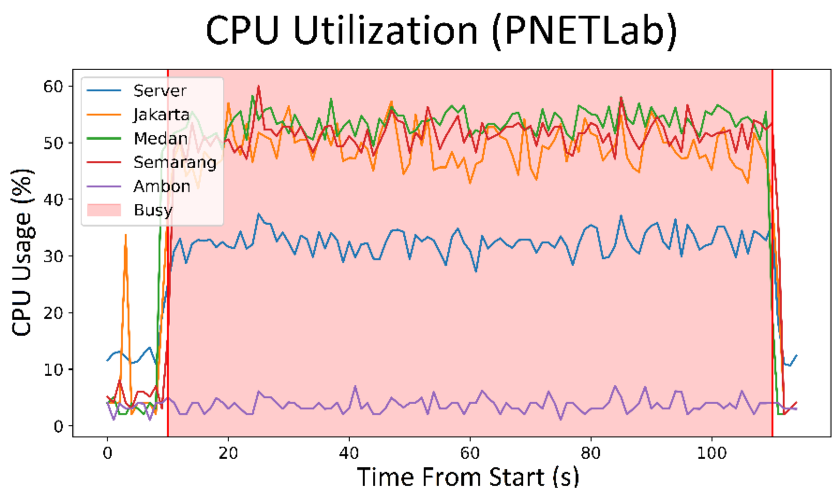
4. Performance evaluations

Using this framework, we can evaluate the parameters of network performance and the hardware performance of NDN nodes. The performance of NDN-IDN scenario using PL compared with NDN-IDN scenario using MiniNDN, as widely used NDN-based emulation. In many research using MN emulation, the researchers did not analyze the CPU and RAM utilization of the scenario. Our research elaborates on why CPU and RAM utilization was impossible to analyze using MN. We also show opportunities with our proposed method to analyze the use of hardware resources in the future.

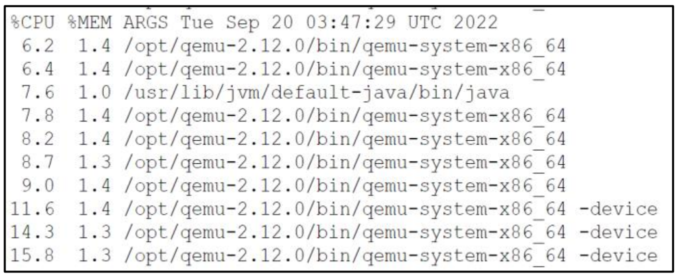
This research uses a virtual machine on commodity hardware, AMD Ryzen 3900x 24 CPU and 80 GB RAM. We allocate a virtual host server with 16 CPUs and 16 GB RAM. Each NDN node is assigned 1 CPU and 2048 MB RAM to check how NDN nodes work for minimal hardware resources. PNETLab and Mini-NDN both have the same virtual host resource allocation, so we can compare how both methods respond in running the scenario.

In our proposed PL system, each node has all independent NDN components, libraries, core components, NDN tools, and interface allocation (depending on the connection needed). All of the NDN components are running on Qemu, then the NDN nodes in the PL system work as a full system

Received: October 12, 2022. Revised: November 14, 2022.

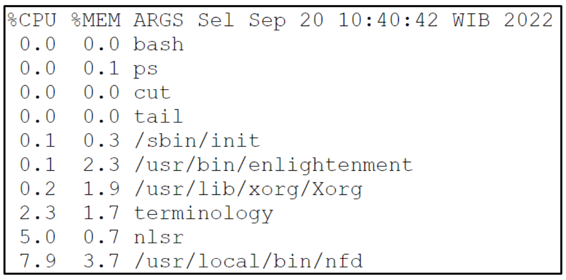


(a)



```
%CPU %MEM ARGS Tue Sep 20 03:47:29 UTC 2022
 6.2  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64
 6.4  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64
 7.6  1.0 /usr/lib/jvm/default-java/bin/java
 7.8  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64
 8.2  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64
 8.7  1.3 /opt/qemu-2.12.0/bin/qemu-system-x86_64
 9.0  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64
11.6  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64 -device
14.3  1.3 /opt/qemu-2.12.0/bin/qemu-system-x86_64 -device
15.8  1.3 /opt/qemu-2.12.0/bin/qemu-system-x86_64 -device
```

(b)



```
%CPU %MEM ARGS Sel Sep 20 10:40:42 WIB 2022
 0.0  0.0 bash
 0.0  0.1 ps
 0.0  0.0 cut
 0.0  0.0 tail
 0.1  0.3 /sbin/init
 0.1  2.3 /usr/bin/enlightenment
 0.2  1.9 /usr/lib/xorg/Xorg
 2.3  1.7 terminology
 5.0  0.7 nlsr
 7.9  3.7 /usr/local/bin/nfd
```

(c)

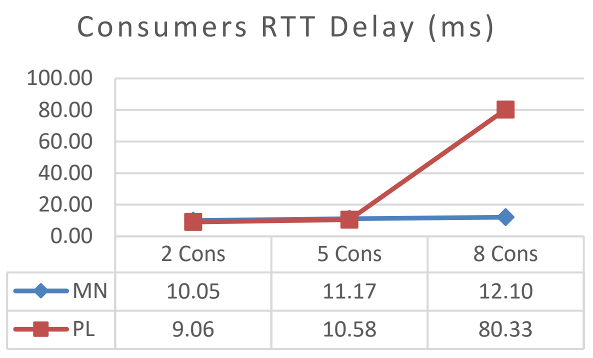
Figure. 7: (a) NDN nodes & host server CPU Utilization on PNETLab, (b) Process CPU & memory usage capture for NDN host server, and (c) Process CPU & memory usage capture for NDN nodes

Table 2. PNETLab CPU, RAM, VRAM record

Time	% CPU used	RAM used (Bytes)	VRAM used (Bytes)
5:14:33 PM	5.376344	655519744	655519744
5:14:34 PM	2.040816	655519744	655519744
5:14:35 PM	4	655765504	655765504
5:14:36 PM	36.458333	720560128	720560128
5:14:37 PM	22.222222	754647040	754647040
5:14:38 PM	4.040404	754647040	754647040
5:14:39 PM	3.092784	754647040	754647040
5:14:40 PM	4.081633	754606080	754606080
5:14:41 PM	3.061224	754589696	754589696
5:14:42 PM	4.081633	755380224	755380224
5:14:43 PM	12.5	756912128	756912128
5:14:44 PM	5.050505	756871168	756871168

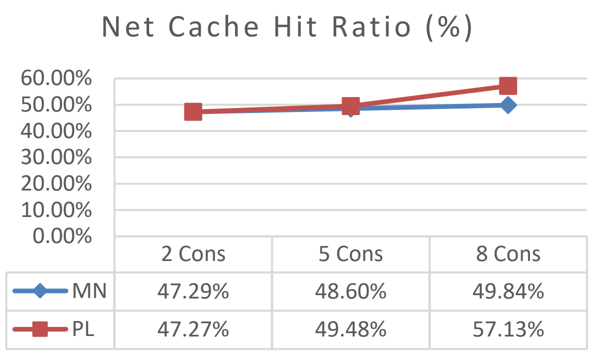
While MN systems use Linux network name-space features to differentiate the process and interfaces for emulating NDN nodes. In the CPU Utilization observation, the difference in emulation methods from PL and MN is very significant. MN system process containerizing leads each process to run on the host server OS. Consequently, we cannot see the detail of what processes occur on each NDN node using MN. We try to capture the processes that occur on each NDN node (consumer, producer, and intermediate NDN nodes) and a host server to get the total load on each hardware resource. It was found that although the processes captured on the node through the Linux X-Term windows in MN, the list of processes obtained was identical for all nodes and host servers, as shown in Fig. 6(a) and 6(b), even though each node has a different activity.

Received: October 12, 2022. Revised: November 14, 2022.



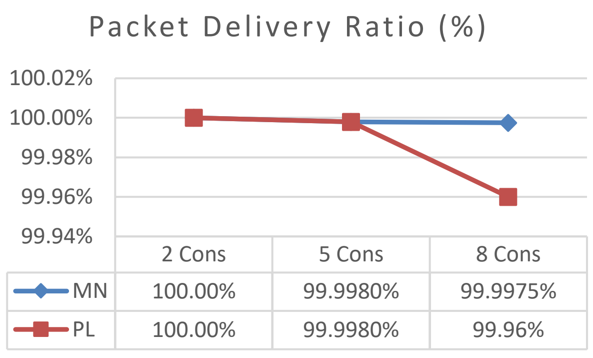
	2 Cons	5 Cons	8 Cons
MN	10.05	11.17	12.10
PL	9.06	10.58	80.33

(a)

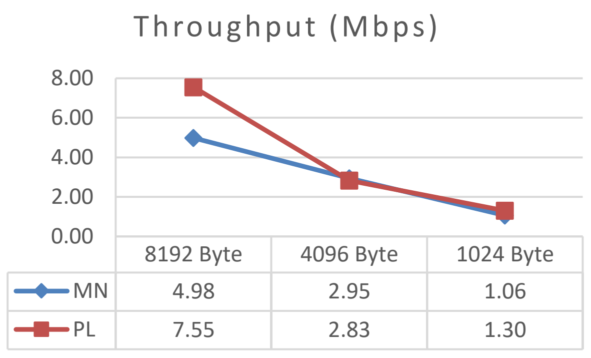


	2 Cons	5 Cons	8 Cons
MN	47.29%	48.60%	49.84%
PL	47.27%	49.48%	57.13%

(c)



	2 Cons	5 Cons	8 Cons
MN	100.00%	99.9980%	99.9975%
PL	100.00%	99.9980%	99.96%



	8192 Byte	4096 Byte	1024 Byte
MN	4.98	2.95	1.06
PL	7.55	2.83	1.30

(b) (d)

Figure. 8 Experiment result: (a) Consummers RTT delay, (b) packet delivery ratio (PDR), (c) Network cache hit tatio (nCHR), and (d) throughput

Whereas in PL, if we access the NDN node, we can catch the different processes between these nodes. Qemu complete system virtualization provides isolation of processes running on each NDN node, depicted in Fig. 7(b) and 7(c). Fig. 7(a) shows that the Jakarta (producer), Semarang, and Medan (consumer) nodes have the highest CPU load. The Ambon node, which only acts as an intermediate node, has the lowest CPU load. Therefore, in the PL emulation environment, we could see the detailed process running in every NDN node and the utilization of the process for hardware resources.

Measurements in PL for RAM usage fluctuates and do not show the direct impact of the scenario we built. It's because the RAM is the only temporal buffer for the process running in the NDN node. However, as seen in Table 2, the RAM usage data record shows that all NDN nodes in PL operate below 2048 MB RAM allocation. This means that all NDN processes are not buffered for a long time to be processed by the CPU.

Observations on the round trip (RTT) delay with changes in the number of consumers indicate that more consumers will increase the delay in data communication on the NDN-IDN network, depicted in Fig. 8(a). RTT delay increases due to the increasing number of requests being served on the network. Each consumer generates 100 interest/second, which means 800 interest/second total for 8 (eight) consumers.

The producer responds to the first request from consumer, while the intermediate node cache serves repeated requests.

As shown in Fig. 8(a), a significant delay occurs when eight consumers make requests simultaneously due to the queue to serve all requests. NDN emulation based on PL has a similar result to MN emulation when 2 (two) consumers and 5 (five) consumers generate interest traffic. Nevertheless, NDN emulation on PL suffers high RTT delay in simultaneous 8 (eight) consumer traffic generation. RTT delay increases 8 (eight) times to 80,33 ms.

The significant performance decrease for PL in the 8 (eight) consumers scenario is caused by limited resource allocation on NDN nodes. Each NDN node gets an allocation of 1 (one) CPU core and 2048 MB RAM which can be said to be a minimal resource for network devices. On the other hand, resource restrictions cannot be applied to the MN, so the MN uses all of the host server's resources. The RTT delay will increase significantly when the IDN-NDN

Received: October 12, 2022. Revised: November 14, 2022.

network is given a high load with 8 consumers (generates a total of 800 interests/second). This is caused by a good isolation of the processes running on each NDN node in the PL environment.

When the RTT delay increases significantly, on the other hand, the net Network cache hit ratio (nCHR) in PL tends to give better performance at 8 (eight) consumer scenarios depicting Fig 8(c). Observations on the Network cache hit ratio (nCHR) with changes in the number of consumers indicate that a consumer number increase will also increase the probability of a cache hit. Terms of nCHR mean that the request that appears is not forwarded to the producer to be served. An intermediate node serves the request on the network. With a high nCHR value, the burden on producers will be significantly reduced.

However, the increment in nCHR could not compensate total consumer's interest rate, so the packet delivery ratio (PDR) dropped to 99,6% in PL depict in Fig. 8(b). PDR is a ratio comparison between the total packet delivered to all the transmit packets. Giving so much interest/second request load with minimal hardware resources makes the packet queue intense. Consequently, some requests could not be satisfied because packets wait too long in the queue and cannot be handled by hardware/node resources.

NDN-IDN with consumer increases gives more stable results for the MN emulation environment because there is no resource limitation issue. PL gives solid performance for 2 (two) consumers and 5 (five) consumers scenarios where both interest rates are 200 and 500 interest/s, respectively.

Then we evaluate throughput performance on both PL and ML (see Fig. 8(d)). Three packet sizes are used, 8.192, 4.096, and 1.024 Bytes. As the most stable scenario, we occupy 2 (two) consumers with 100 interest/s each. Results show PL and ML give a close performance for 4k and 1k Bytes packet size, but in 8k Bytes, PL gives higher throughput than MN.

This increase is because sending large content sizes requires fewer requests. So, more requests can be served with fewer requests and larger packet sizes. As elaborated in Eq. (13), PS is proportional to the throughput in the non-blocking condition and resource adequacy. Thus, the PL emulation environment gives a more solid throughput than MN for an 8K Bytes packet size.

5. Conclusions

In this research, we proposed a method as a catalyst for accelerating NDN research toward real-world implementation. This method occupies a UNetLab-based virtualization environment for extensive observations for the researcher.

As a corroboration proof of this method, we build a test case in the PNETLab compared with Mini-NDN as a widely used NDN emulation mechanism. The result showed PL method could give an extensive measurement of NDN nodes, while MN failed.

However, PL and MN give similar results in IDN-NDN network performance measurement for 100 to 500 interest/s, 7,59% in RTT delay, and 0,89% in nCHR. It means that both PL and MN can be used to analyze NDN performance. Furthermore, only PL can give detailed data on the hardware utilization process for each node, which is very important for NDN implementation.

6. Future work

For future research, we will use this method to explore NDN-based IoT, which is hardware resource sensitive.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Conceptualization, TAW and NRS; methodology, TAW and NRS; software, TAW and ADR; validation, TAW, HEN, and ADR; formal analysis, TAW and NRS; investigation, TAW, NRS and HEN; resources, TAW; data curation, TAW; writing—original draft preparation, TAW and LVY; writing—review and editing, TAW and LVY; visualization, TAW; supervision, NRS and HEN. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

This work was partially supported by Institut Teknologi Bandung, Telkom University, and Ministry of Research Technology and Higher Education Republic of Indonesia.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content", In: *Proc of the 5th International Conference on Emerging Networking Experiments and Technologies - CoNEXT '09*, p. 1, 2009.

[2] L. Zhang, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking", *ACM SIGCOMM Comput. Commun. Rev.*, Vol. 44, No. 3, pp. 66–73, 2014.

[3] T. Koponen, M. Chawala, B. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica,

"A data-oriented (and beyond) network architecture", *ACM SIGCOMM Comput. Commun. Rev.*, Vol. 37, No. 4, pp. 181–192, Aug. 2007, doi: 10.1145/1282380.1282402.

[4] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further: From PSIRP to PURSUIT", *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng.*, Vol. 66 LNICST, pp. 1–13, 2012.

[5] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of Information (NetInf) – An information-centric networking architecture", *Comput. Commun.*, Vol. 36, No. 7, pp. 721–735, Apr, 2013, doi: 10.1016/j.comcom.2013.01.009.

[6] G. García , A. Beben, F. Ramon, A. Maeso, I. Psaras, G. Pavlou, N. Wang, J. Sliwinski, S. Spirou, S. Soursos, and E. Hadjioannou, "COMET: Content mediator architecture for content-aware networks", In: *Proc. of 2011 Futur. Netw. Mob. Summit, Futur. 2011*, pp. 1–8, 2011.

[7] A. Detti, N. B. Melazzi, S. Salsano, and M. Pomposini, "CONET: A Content Centric Inter-Networking Architecture", In: *Proc of the ACM SIGCOMM Workshop on Information-Centric Networking - ICN '11*, 2011.

[8] N. B. Melazzi, S. Salsano, A. Detti, G. Tropea, L. Chiariglione, A. Difino, A. Anadiotis, A. Mousas, I. Venieris, and C. Patrikakis, "Publish/subscribe over information centric networks: A Standardized approach in CONVERGENCE", In: *Proc. of 2012 Futur. Netw. Mob. Summit, Futur. 2012*, pp. 1–8, 2012.

[9] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "MobilityFirst: A Robust and Trustworthy MobilityCentric Architecture for the Future Internet", *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, Vol. 16, No. 3, p. 2, 2012.

[10] A. Tagami and M. Arumaithurai, "GreenICN Project: Architecture and Applications of Green Information Centric Networking", *IEICE Trans. Commun.*, Vol. E99.B, No. 12, pp. 2470–2476, 2016, doi: 10.1587/transcom.2016CNI0001.

[11] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the Evolution of ndnSIM", *ACM SIGCOMM Comput. Commun. Rev.*, Vol. 47, No. 3, pp. 19–33, 2017, doi: 10.1145/3138808.3138812.

[12] M. Amadeo, G. Ruggeri, C. Campolo, and A. Molinaro, "IoT services allocation at the edge via named data networking: From optimal bounds to practical design", *IEEE Trans. Netw. Serv. Manag.*, Vol. 16, No. 2, pp. 661–674, 2019, doi: 10.1109/TNSM.2019.2900274.

[13] M. Amadeo, C. Campolo, G. Ruggeri, G. Lia, and A. Molinaro, "Caching transient contents in vehicular named data networking: A performance analysis", *Sensors (Switzerland)*, Vol. 20, No. 7, pp. 1–17, 2020, doi: 10.3390/s20071985.

[14] M. S. Budiana, M. M. Nadra, T. R. Hapsari, R. Mayasari, and N. R. Syambas, "Impact of the Content Store Scaling toward the LRU and FIFO Cache Replacements on NDN using Mini-NDN", In: *Proc 15th Int. Conf. Telecommun. Syst. Serv. Appl. TSSA 2021*, 2021.

[15] X. Guo, S. Yang, L. Cao, J. Wang, and Y. Jiang, "A new solution based on optimal link-state routing for named data MANET", *China Commun.*, Vol. 18, No. 4, pp. 213–229, 2021.

[16] D. Saxena and V. Raychoudhury, "Design and Verification of an NDN-Based Safety-Critical Application: A Case Study with Smart Healthcare", *IEEE Trans. Syst. Man, Cybern. Syst.*, Vol. 49, No. 5, pp. 991–1005, 2019, doi: 10.1109/TSMC.2017.2723843.

[17] L. Gameiro, C. Senna, and M. Luís, "Insights from the Experimentation of Named Data Networks in Mobile Wireless Environments", *Futur. Internet*, Vol. 14, No. 7, pp. 1–18, 2022, doi: 10.3390/fi14070196.

[18] A. Friyanto, W. T. Ariefianto, and N. R. Syambas, "Analysis Operation NLSR With Ubuntu as NDN Router", In: *Proc. of 2019 IEEE 5th International Conference on Wireless and Telematics (ICWT)*, 2019.

[19] "EVE-NG Site", https://www.eve-ng.net/ (Accessed Jul. 30, 2022).

[20] "PNETLab : Lab is Simple Site", https://pnetlab.com/pages/main (Accessed Jul. 30, 2022).

[21] "Home - Docker", https://www.docker.com/ (Accessed Jul. 30, 2022).

[22] "GitHub - GNS3/dynamips: Dynamips development", https://github.com/GNS3/dynamips (Accessed Jul. 30, 2022).

[23] "ndn-traffic-generator_modified", https://github.com/aderama2711/ndn-traffic-generator_modified (Accessed Jul. 31, 2022).

[24] L. V. Yovita, N. R. Syambas, I. J. M. Edward, and N. Kamiyama, "Performance analysis of cache based on popularity and class in named data network", *Futur. Internet*, Vol. 12, No. 12, pp. 1–22, 2020, doi: 10.3390/fi12120227.

[25] T. A. Wibowo, N. R. Syambas, and H. Hendrawan, "The routing protocol efficiency of named data network", In: *Proc. of 2020 14th International Conference on*

Telecommunication Systems, Services, and Applications (TSSA), pp. 1-5, 2020, doi: 10.1109/TSSA51342.2020.9310886.