Comparison of Caching Replacement Policies in Changing the Number of Interest Packets on Named Data Networks Using Mininet-NDN

Faishal Zharfan
School of Electrical Eng. & Informatics
Bandung Institute of Technology
Bandung, Indonesia
18119002@telecom.stei.itb.ac.id

Larastya Devindira Hasnaa
School of Electrical Eng. & Informatics
Bandung Institute of Technology
Bandung, Indonesia
18119022@std.stei.itb.ac.id

Nana Rachmana Syambas
School of Electrical Eng. & Informatics
Bandung Institute of Technology
Bandung, Indonesia
nana@stei.itb.ac.id

[1,2]Ridha Muldina Negara
[1]*School of Electrical Eng.& Informatics*
Bandung Institute of Technology
[2]*School of Electrical Engineering*
Telkom University
Bandung, Indonesia
ridhanegara@telkomuniversity.ac.id

2021 15th International Conference on Telecommunication Systems, Services, and Applications (TSSA) | 978-1-6654-2892-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/TSSA52866.2021.9768267

***Abstract*— Named Data Networking (NDN) is a new concept of network architecture. Unlike TCP/IP which is based on IP addresses and each IP layer packet needs to travel end to end from source to destination address to get the requested content, NDN allows the process of requesting content to be a shorter journey due to a component called caching. With caching, the Content Store (CS) on the NDN router will be populated with the content requested by the consumer. This content retention mechanism is managed using caching policies, which are of two types, cache placement policies, and cache replacement policies. This study examines the performance of FIFO and LRU cache replacement policies for a certain number of interest packages based on parameters such as delay, packet loss, and throughput. This study uses nodes from the pan-European GEANT network topology in Mini-NDN, a Mininet-based network emulator for NDN. Tests using emulators provide research results that are close to real-world conditions. Based on the experiment, LRU has better performance than FIFO in handling the increase in data interest as seen from throughput and packet loss. However, in terms of average delay, FIFO is better than LRU.**

***Keywords*—NDN, Cache Replacement Policy, FIFO, LRU, Mini-NDN**

I. Introduction

Today’s internet connection is heavily based on TCP/IP protocol stack. With TCP/IP, communication over the internet between hosts is held as end-to-end communication. It means that each IP layer packet that was sent from source to destination must contain the source address and the destination address to support host-to-host communication. However, packets that are currently sent all over the internet to request searching for information are getting large in numbers. Hence, there is an increase in internet traffic [1], [2].

Another important thing that we have to realize, what people value on the internet is the content or information itself. Or, we can say that it has shifted from ‘where’ the information data is located to ‘what’ is the information contained from the data. Another change is the way people use to exchange information on internet services. Now, most internet services are conducted by sharing content from one user to other hosts or so-called resource sharing. Thus, having a lot of end-to-end traffic from source to destination within the IP protocol is not efficient anymore, and a new internet architecture to access content efficiently is needed [1], [2].

A brand new concept called Content-Centric Networking (CCN) is introduced in [3]. This concept built communication architecture based on named data. According to Jacobson in his proposal about CCN [3], CCN maintains IP protocol's simplicity and scalability, while offering much better security, increasing efficiency in delivery, also tolerating disruption. Derived from CCN, there is a concept called Named Data Networking (NDN), which transmits data communication based on identity used by content name. Some studies have also shown a convinced enough result that NDN is way better than IP protocol in terms of efficiency as requests are addressed to specific content that they want, not to the specific IP address of content producer anymore. Therefore, users can just focus on requesting what data that they want [4].

NDN has various components, such as Naming, Caching, Security, Routing, and Forwarding [5]. As a future replacement of current internet architecture, a comparison between NDN and IP architecture in hourglass shape is shown below in Fig. 1.

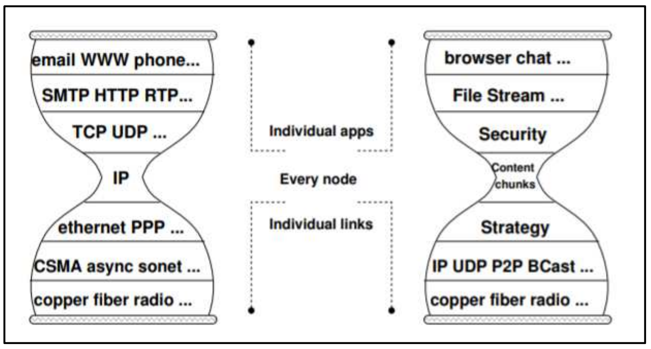


Fig. 1. Hourglass shape architecture comparison between TCP/IP and NDN[3].

According to Fig. 1, NDN relies heavily on content chunks. To store the content, data is placed on nodes of the NDN Router. Through the cache placement policy, data will be placed on the content store of NDN routers that are located the closest to the consumer [6]. When the capacity of the content store is already full, the NDN router will use caching replacement policy to replace previous content with the new ones.

Consequently, this research is done to study the effect of using particular caching replacement policies, in this case, LRU or FIFO towards interest packet, from the point of view of the delay, packet loss, and throughput in mini-NDN. Results of this study will be very useful to give considerations when choosing the perfect caching replacement policy between LRU and FIFO when performing caching replacement process for a certain number of packet interests.

II. Named Data Networking

Internet current usage examples are social media networking, e-commerce, digital media, and applications for smart devices. These kinds of applications belong to a resource-sharing type of use that is mentioned previously. As a result, the internet is mainly used as a distribution network. This condition made NDN become a perfect concept to implement as NDN itself has a paradigm of a distribution network that concentrated on the content chunk. Based on Fig. 1, the concept of NDN evolves in the thin waist of content chunks to enable the creation of a distribution network [7]. To do that, two types of packets will be used in process of requesting the content and getting the content itself.

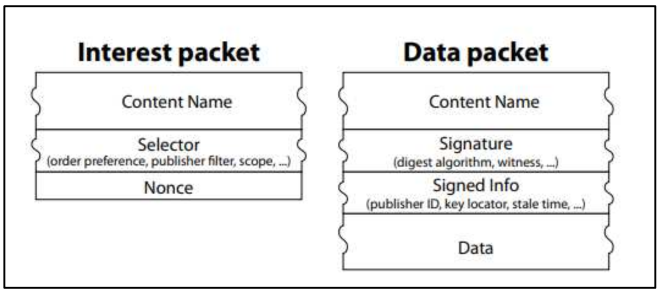

Interest packet
Content Name
Selector
(order preference, publisher filter, scope, ...)
Nonce
Data packet
Content Name
Signature
(digest algorithm, witness, ...)
Signed Info
(publisher ID, key locator, stale time, ...)
Data

Fig. 2. Types of a packet in NDN [3].

As a part of CCN, NDN has 2 types of packets structure, which are interest and data as shown in Fig. 2. Interest packet is used when a consumer requests the content. Interest packet will be broadcast all over the network, so if any node hears the interest and has the content consumer was looking for, that node can respond with a data packet. Both interest and data packets use names to identify the exchanged content. Names that are used in NDN do not need to be in the same context as the content, but one thing for sure is this naming scheme must be hierarchical so that the prefix match of data is equivalent with the name from subtree specified by the interest packet [3].

We have stated previously that NDN has a paradigm of distribution network. Loads from the data and content server are distributed into nodes in NDN routers. NDN router has 3 components, which consist of Content Store (CS), Pending Interest Table (PIT), and Forwarding Information Base (FIB) [4].

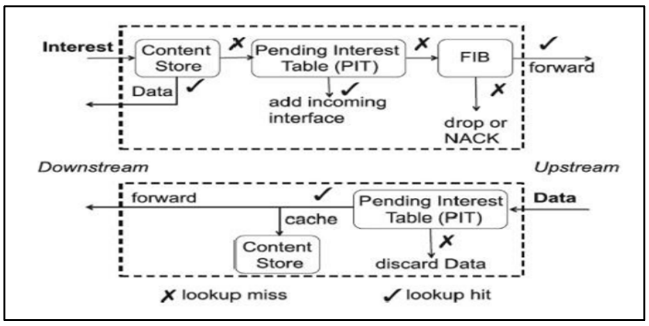


Fig. 3. Process in NDN Router [8].

According to Fig. 3, the way NDN router works is begun by interest packet who came to NDN router. Information from interest will be kept in PIT and it will be used as a hint for data packet transmission path when it goes back to the consumer from the producer. But, if the requested data is already available on CS, data will be sent back to the consumer based on the information in PIT. For cases like the data that are not available in the node, the requested data will be searched in the FIB table, which owns information of the next hop where the data will be forwarded next [4].

For NDN router to store the content, each of them must be equipped with a buffer, along with using the most suitable caching policy for the content that will be selected to store, content that will be deleted if the buffer is already full, or which nodes that will be used to store the content [4].

Therefore, caching policy has a very important role in the network's performance. Caching policy itself is divided into two types, caching placement policy and caching replacement policy [9]. In this research, we are going to examine the cache replacement policy only. Examples of cache replacement policies that will be used in the NDN Forwarding Daemon of our simulation are Least Recently Used (LRU) and First-In-First-Out (FIFO).

LRU as caching replacement policy means that the least recently accessed content will be discarded when the cache is over maximum size. But, if the cache size is not full yet, cache memory will keep the object [8]. It uses recent information to increase the cache hit ratio without considering the popularity of the content. So, even if people are less interested in the content, LRU will still replace it with the most recent one [10] without considering when was the object stored for the first time [11]. This timestamp-based policy is said to be popular in use because it has been known to perform well. It has a complexity of O(1) [9] and is also capable of storing the most recent data for a longer time, thus it will increase the success rate of a cache hit [8].

On the other hand, FIFO as a cache replacement policy will replace the oldest content with the newly arrived one without considering when was the last time the content was accessed. FIFO has an assumption that the oldest content is not going to be requested anymore. Hence, it will delete the earliest content and may cause miss and delay as the deleted content might still be in use. However, FIFO still has its advantage that is efficiency in the cache replacement process because its queue system is based on a sequence of the stored objects. While FIFO is static and easy to combine with other policies, FIFO also rarely encounters a pile of a stack [11].

FIFO and LRU have their advantage and disadvantage. Both FIFO and LRU have similarities in that they will cache most of the received content regardless of the popularity of the content. The idea of combining both policies as proposed in [11] will result in maximum hit rate, minimum miss rate, and efficient cache usage. Choosing one of them and implementing it in a suitable scenario will deliver a good result as well [11], [12].

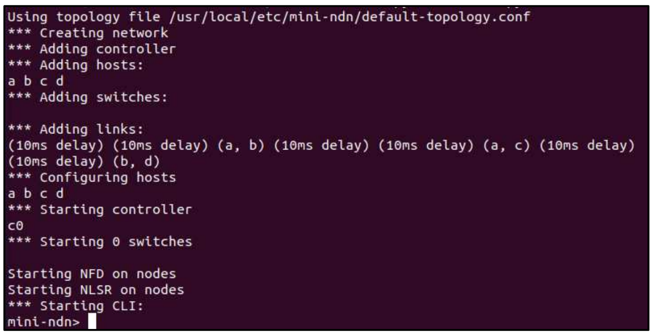
III. Research Scenario

In this research, simulations are performed to find out the effect of choosing a certain cache replacement policy toward traffic parameters of interest packet size and conducted using mini-NDN, a Mininet-based network emulator for NDN. Simulations were performed in several chosen nodes of the GEANT topology of the pan-European network that is representative enough to become a content producer and consumer.

A. Mini-NDN

In this research, simulation is performed in mini-NDN, network emulator for NDN which is based on Mininet. Mininet itself is can emulate a full set network of hosts, links, and switches on a single machine. As an emulator, mini-NDN is scalable in that capable of running actual instances of NFD and NLSR. Mini-NDN uses the NDN libraries, NFD, NLSR, and tools released by the NDN project on a single system [13]–[15].

We will run the mini-NDN emulator on OS Ubuntu and some early configurations and downloads will be needed to do so. After that, we can start running mini-NDN in a mini-NDN folder. With the command '`sudo python2 mnndn.py`', we can run mini-NDN which uses default topology as shown in Fig. 4 below.



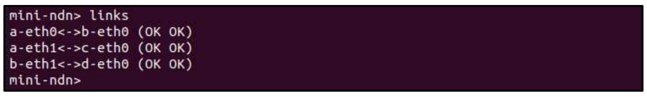
```
Using topology file /usr/local/etc/mini-ndn/default-topology.conf
*** Creating network
*** Adding controller
*** Adding hosts:
a b c d
*** Adding switches:

*** Adding links:
(10ms delay) (10ms delay) (a, b) (10ms delay) (10ms delay) (a, c) (10ms delay)
(10ms delay) (b, d)
*** Configuring hosts
a b c d
*** Starting controller
c0
*** Starting 0 switches

Starting NFD on nodes
Starting NLSR on nodes
*** Starting CLI:
mini-ndn>
```

Fig. 4. Process of start running mini-NDN

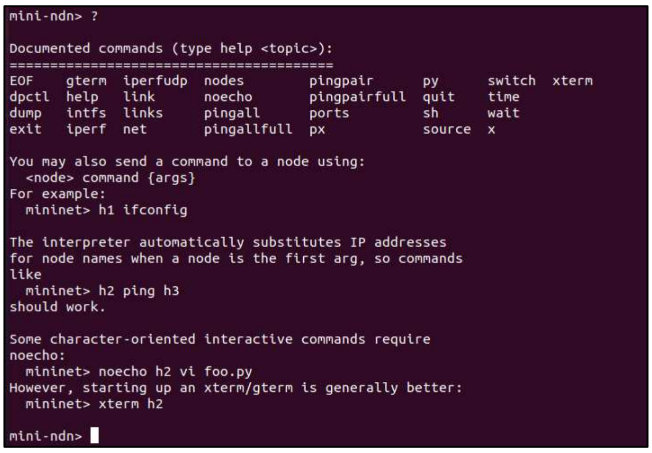
To find out about connected nodes in this default topology, we can type '`links`' as a command as shown in Fig. 5 below.



```
mini-ndn> links
a-eth0<->b-eth0 (OK OK)
a-eth1<->c-eth0 (OK OK)
b-eth1<->d-eth0 (OK OK)
mini-ndn>
```

Fig. 5. Links of nodes in default topology di mini-NDN

To discover more commands that we can do to our nodes in mini-NDN, we can use '?' as a command to check out other available commands as shown in Fig. 6.



```
mini-ndn> ?

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch  xterm
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      wait
exit   iperf  net       pingallfull  px            source  x

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2

mini-ndn>
```

Fig. 6. Available commands in mini-NDN

Compared to ndnSIM, an NS-3 based simulator for NDN and is implemented as a new network-layer protocol model [16], ndnSIM still has some drawbacks such as the need to do modifications in some parts for real-world applications, the version of NFD which may not be the latest one, and the limitation to interact directly in a real-time scenario which made visualization possible [14]. As mini-NDN was made to solve those problems, therefore, we can say that mini-NDN is better than ndnSIM.

B. GEANT Topology Scenario

GEANT is a pan-European topology connect research and education network in Europe's National Research and Education Networks (NRENs). Illustration of GEANT topology itself. A picture of how does GEANT topology covers 65 countries beyond Europe is depicted below [17].

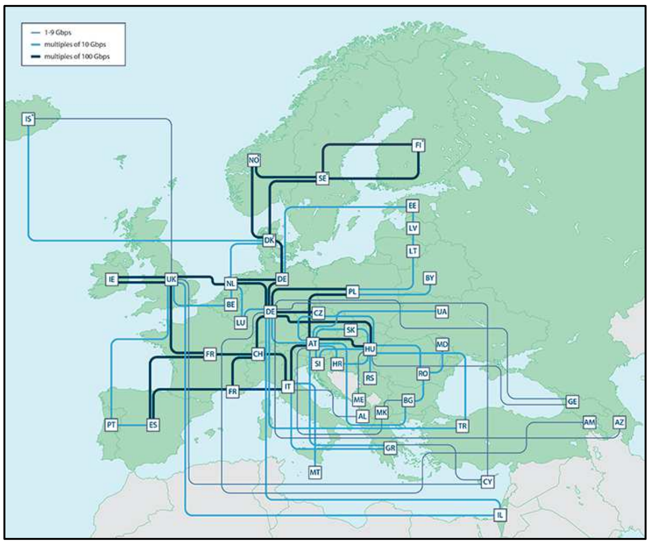


Fig. 7. GEANT Topology [17].

GEANT topology connects more than 50 million users at 10,000 institutions across the European continent and maintains itself as the most advanced and best-connected research and education network in the world. This topology connects nodes from numerous European countries, such as Norway, the United Kingdom, Greece, Spain, Germany, Belgium, Italy, Austria, and many more [17].

Based on network topology in Fig. 7, we chose node MD or Moldova as a content producer, also node CY or Cyprus, and node IR or Ireland as consumers. As the experiment is going to be done to examine caching replacement policy, so the content store will be made into full in terms of capacity. Only after that, data retrieval from the experiment would begin with consumers requesting certain content to the network, and we could start to find out about the resulting traffic parameters, such as delay, packet loss, and throughput for many interest packets.

IV. Simulation Results

To conduct this research, we combined Mini-NDN with a traffic generator. We will begin by determining prefixes that we want to use. In this case, we used 15 prefixes, and each prefix has a different weight. Then, we will make configurations for the server and client for that 15 prefixes, such as 100 seconds for freshness, 50 bytes for size of content, and the rest are set to default. After that, the capacity of the content store will be set to 10. Until here, the configuration is finished. An automation script will be made afterward so that an empty CS would be filled with dummy packets, then continued with starting a simulation until the result is out and will be saved in a txt-format file. After each request is done, CS will be deleted and other dummy packets are going to fill the Content Store. This process is going to be repeated until it is over.

A. *Priority-FIFO*

FIFO cache replacement works by replacing the oldest content with the newly arrived data. In this experiment, delay, throughput, and packet loss measurements will be performed on each number of data interests. Firstly, '`nfd.py`' will be configured by changing cache policy to '`priority_fifo`' and setting the capacity to 10. Consequently, on Mini-NDN, since we are trying to examine the behavior of certain cache replacements, therefore the Content Store (CS) on every node should be filled by dummy data packets and should be cleared after every request is over. In the end, requests will be performed 10 times using different interests. The previous step is performed automatically using a shell script. After the simulation has finished, the results can be seen on the node's folder
'`/tmp/minindn/<node>/result<interest>.txt`'.

From those results, we obtained two parameters which are delay information and packet loss. For easier reading, the information will be visualized using MATLAB.

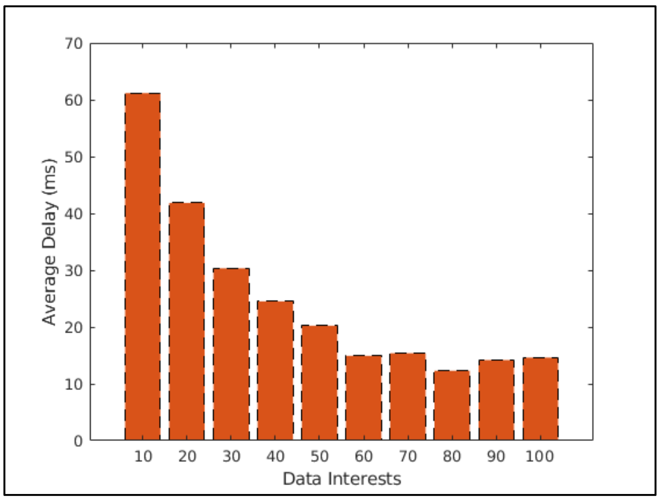


Fig. 8. Average Delay resulted from data interest that was sent

As for the delay, we measured by using the average RTT value on every node of every interest. As can be seen in Fig. 8 above, it can be known that the delay is relatively decreasing as the data interests rise and interest 10 seems to be the peak. This happened because as the request rises, the probability of getting the same prefix request will increase, therefore the average delay will decrease.

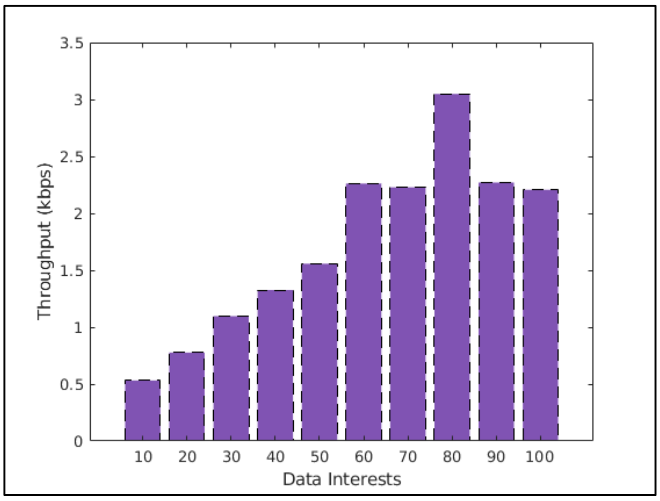


Fig. 9. Throughput resulted from data interest that was sent

Throughput is defined as total data that can be sent at certain times. In this measurement, we defined throughput on a kbps scale. As can be seen in Fig. 9 above, the throughput is relatively increasing as the data interest increases and interest 80 seems to be the peak. This happened because the average delay (RTT) and throughput are inversely proportional.

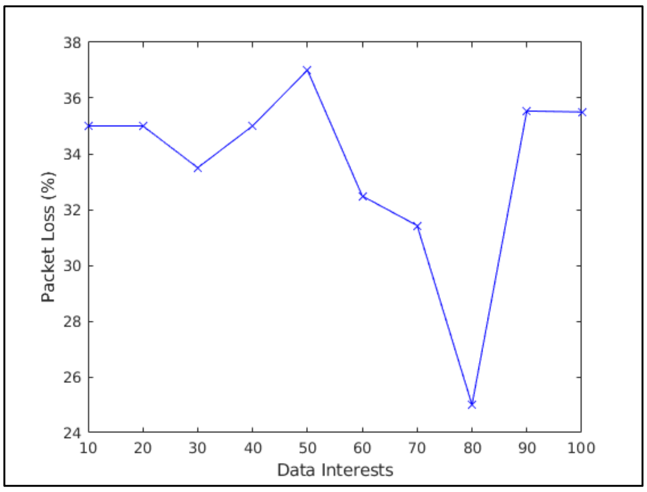


Fig. 10. Percentage of packet loss resulted from data interest that was sent

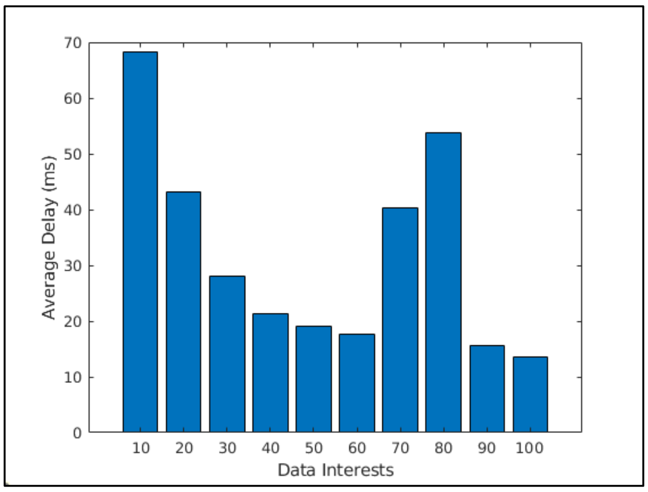


Fig. 11. Average Delay resulted from data interest that was sent

As for the packet loss, it can be seen in Fig. 10 above. The packet loss seems to decrease for a while as the data interest increases. This happened because we are using a traffic generator that might mimic the real network traffic and the generator only chooses certain prefixes out of 15 randomly on default conditions.

B. LRU

LRU cache replacement works with replacing the least used data with the newly arrived data. As before, in this experiment, delay, throughput, and packet loss measurements will be performed on each data interest. Firstly, `nfd.py` will be configured by changing cache policy to '`lru`' and setting the capacity to 10, just like before. Consequently, on Mini-NDN, since we are trying to examine the behavior of certain cache replacements, therefore the Content Store on every node should be filled by dummy data packets and should be cleared after every request is over. In the end, requests will be performed 10 times using different interests. The previous step is performed automatically using a shell script. After the simulation has finished, the results can be seen on the node's folder '`/tmp/minindn/<node>/result<interest>.txt`'.

From those results, we obtained two parameters which are delay information and packet loss. For easier reading, the information will be visualized using MATLAB.

As for the delay, just like before, we measured by using the average value on every node on every interest. It can be seen in Fig. 11 above that the delay is relatively decreasing as the data interest rises and interest 10 seems to be the peak. This happened just like the previous experiment that is as the request rises, the probability of getting the same prefix request will increase, and thus the average delay will decrease.

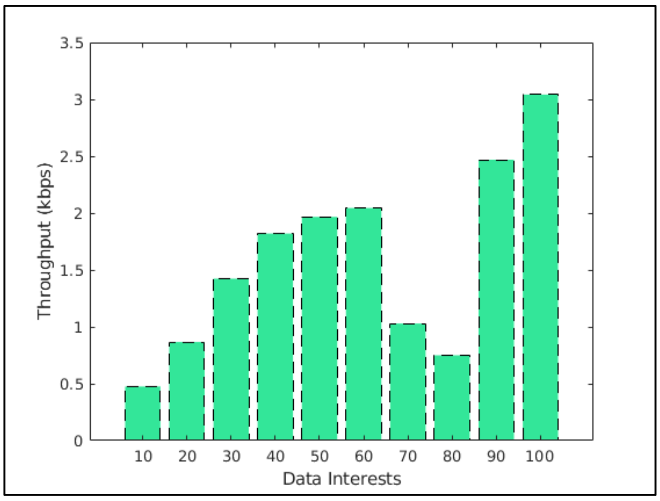


Fig. 12. Throughput resulted from data interest that was sent

As for the throughput, we performed the same measurement as the previous experiment did. It can be seen in Fig. 12 above, the throughput is relatively increasing as the data interest increases and interest 10 seems to be the minimum. This happened because the average delay (RTT) and throughput are inversely proportional, thus the maximum value in delay will be the minimum value in throughput.

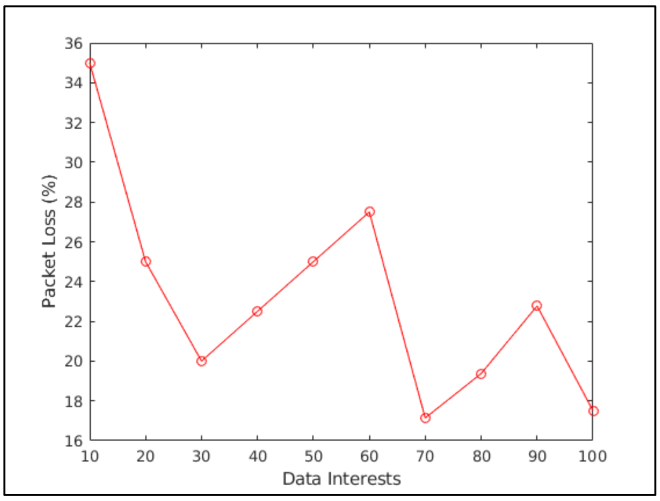


Fig. 13. Percentage of packet loss resulted from data interest that was sent

As for the packet loss, it can be seen in Fig. 13 above. The packet loss seems to be decreasing as the data interest increases. This happened because we are using a traffic generator that might mimic the real network traffic and the generator only chooses certain prefixes out of 15 randomly on default conditions.

V. Evaluation and Analysis

After we performed both experiments, the following step is to compare the performances between two caching policies that were used in this experiment.

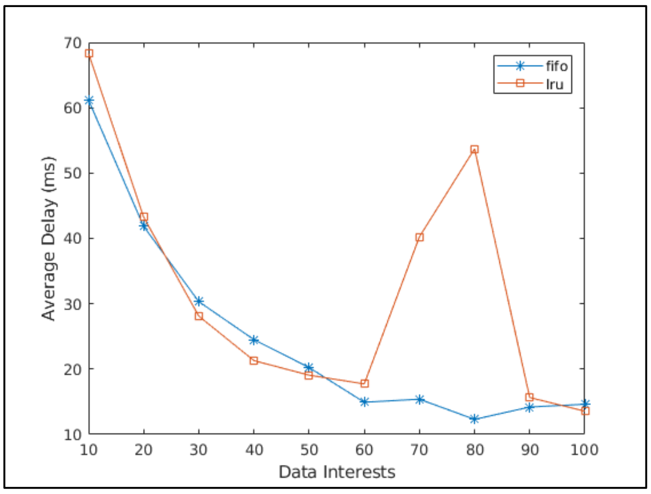


Fig. 14. Comparison of resulted average delay between FIFO and LRU

The picture above, Fig. 14 depicts the comparison between LRU and FIFO in terms of delay. From the picture, it can be known that both policies are similar.

At first, the average delay of FIFO is lower than LRU, especially when the number of data interest is around 10 until 20. Then, when the number of data interest is increased to 30, there is a point where both policies have the same average delay score of around 35 - 40 milliseconds.

After that, the situation is reversed. The average delay of FIFO started to become higher than LRU until it reached a point between the number of data interest is around 50 until 60 where both policies have the same average delay score of nearly 20 milliseconds. Then, the situation is turned again. The average delay of LRU starts to become higher than FIFO and continues to increase rapidly until the average delay score reaches nearly 55 milliseconds where the number of data interest is equal to 80.

After that, it drops down rapidly as well and continues to stabilize itself until both policies have the same score of average delay of around 15 milliseconds where the number of data interest is around 95. The situation is flipped over again afterward. That is why more experiments to increase data interest are still needed to have a better conclusion on which policy will have the lower average delay based on the comparison graph of average delay between FIFO and LRU.

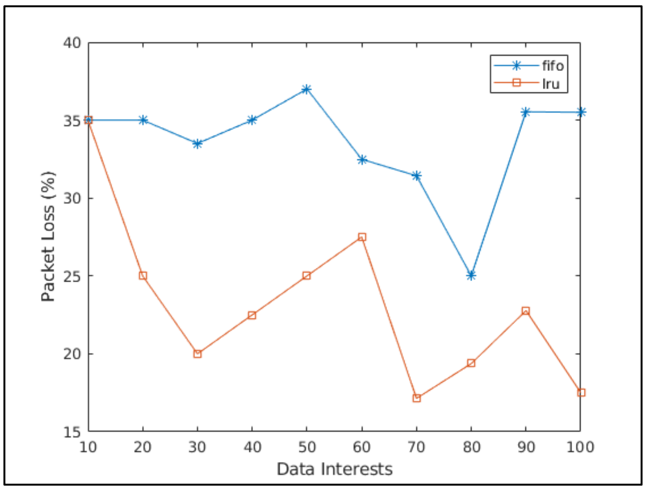


Fig. 15. Comparison of resulted packet loss between FIFO and LRU

Fig. 15 above depicts the comparison between packet loss in LRU and FIFO. The packet loss is primarily affected by the Content Store (CS), waiting time, and data interest rate. A larger Content Store (CS) leads to lower packet loss due to lower waiting time. Waiting time or known as PIT lifetime is defined as the maximum time of the router waiting for the request to be obtained before the request is dropped. Higher data interest will relatively lead to lower packet loss if and only if the spread amount of a particular request is not equal. Longer waiting time tends to lower packet loss and on the other hand, lower waiting time leads to higher packet loss.

Initially, both of them have packet loss with a percentage of 35%. As the number of data interest continues to increase, the packet loss percentage of FIFO stays higher compared to LRU. At some points, the percentage of packet loss in LRU does not even reach 20%. It happens when the number of data interests is 70 and 90.

Therefore, it can be known that LRU is constantly better in terms of packet loss. This happens because as we have known from the delay that LRU has better performance and that can be linearly proportional to the packet loss. However, the expired time of waiting also takes control of the packet loss.

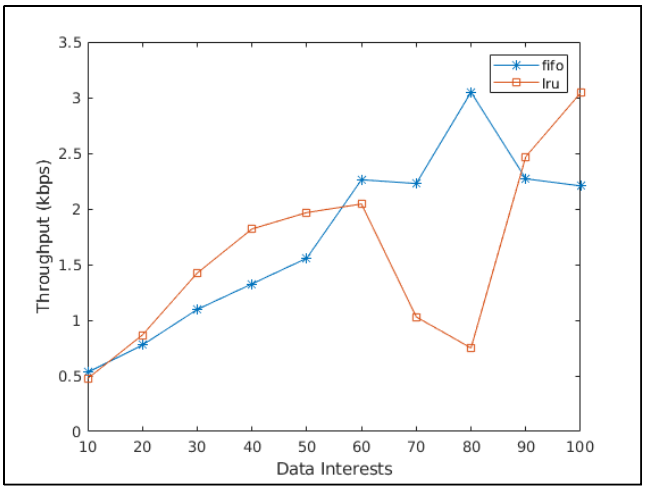


Fig. 16. Comparison of resulted throughput between FIFO and LRU

Fig. 16 above depicts the comparison between the throughput of LRU and FIFO. The throughput is primarily affected by the packet loss. Higher packet loss will lead to lower throughput.

Initially, FIFO has a slightly higher throughput, then at some point between 10 - 20 data interest, a throughput of between 0.5 and 1 kbps is reached for both LRU and FIFO. After that, the situation is reversed as LRU has a better throughput until the packet of data interest is between 50 - 60. At this point, both LRU and FIFO have the same throughput of around 2 kbps.

Then, the situation is reversed. FIFO has a better throughput as it gets higher than LRU with a peak of 3 kbps when the number of data interest is 80. At the same time, an opposite situation occurs to LRU, the throughput is dropping down to between 0.5 and 1 kbps. Subsequently, LRU bounces back up and FIFO goes down a little until both of them have the same throughput of approximately 2.5 ms when the data interest is a little less than 90. From this point, the throughput of LRU continues to go up and as for FIFO, the throughput is slightly going down.

Throughput has a linear proportional relationship with packet loss. Based on Fig. 15, we conclude that LRU is better as it has a lower percentage of packet loss compared to FIFO. Lower packet loss means that more data are being successfully transferred to the determined destination which leads to having a higher throughput. Consequently, we could say that LRU has a higher throughput than FIFO even if the information that we could get from Fig. 16 is still a little unclear and further experiments will be needed.

TABLE I. COMPARISON OF AVERAGE SCORE OF PARAMETERS ON LRU AND FIFO

	Delay (millisecond)		Packet Loss (%)		Throughput (kbps)	
	FIFO	**LRU**	**FIFO**	**LRU**	**FIFO**	**LRU**
Average	24.950	32.093	33.545	23.179	1.7303	1.588

According to table 1 which shows comparisons of parameter scores between LRU and FIFO, it can be seen the average result of each parameter.

VI. Conclusion and Future Work

The results of the Comparison of Caching Replacement Policies in Changes in the Number of Interest Packages show different performances between FIFO and LRU.

In terms of delay, FIFO is better because it has a lower delay than LRU, so we can conclude that FIFO is better than LRU. However, the result may not satisfy the theory since the experiment we conducted only performed 10 variations of interest, and the difference between each interest is 10. It is recommended to choose more variations in terms of data interests and the difference between data interests should be more distant to obtain better results.

Regarding packet loss, LRU has a lower percentage compared to FIFO. Thus, we can say that LRU is better than FIFO in terms of packet loss. To get better results, we suggest setting the PIT lifetime longer than the default conditions to prevent packet loss during transmission. A lower percentage of packet loss will lead to higher throughput as it means that more data is being successfully transferred. Hence, in the case of throughput, LRU will also be considered better than FIFO. Since LRU shows better results in 2 out of 3 parameters, we will conclude that LRU is a better caching replacement policy compared to FIFO.

References

[1] H. Situmorang, N. R. Syambas, and T. Juhana, "The Effect of Scaling The Size of Topology and Content Stored on The Named Data Networking," *Proceeding 2016 10th Int. Conf. Telecommun. Syst. Serv. Appl. TSSA 2016 Spec. Issue Radar Technol.*, pp. 1–6, 2017, doi: 10.1109/TSSA.2016.7871110.

[2] P. Singh, R. Kumar, S. Kannaujia, and N. Sarma, "Adaptive Replacement Cache Policy in Named Data Networking," *2021 Int. Conf. Intell. Technol. CONIT 2021*, pp. 1–5, 2021, doi: 10.1109/CONIT51480.2021.9498489.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking Named Content," *Commun. ACM*, vol. 55, no. 1, pp. 117–124, 2012, doi: 10.1145/2063176.2063204.

[4] L. V. Yovita and N. R. Syambas, "Content Storage Effect on The Named Data Network Traffic Load," *Proceeding 2017 11th Int. Conf. Telecommun. Syst. Serv. Appl. TSSA 2017*, vol. 2018-Janua, pp. 1–5, 2018, doi: 10.1109/TSSA.2017.8272941.

[5] Anjali, "Components of Named Data Networking," *Int. J. Innov. Eng. Technol.*, vol. 7, no. 3, pp. 543–552, 2016.

[6] R. M. Negara and N. Rachmana Syambas, "Caching and machine learning integration methods on named data network: A survey," 2020, doi: 10.1109/TSSA51342.2020.9310811.

[7] N. R. Syambas, H. Tatimma, A. Mustafa, and F. Pratama, "Performance Comparison of Named Data and IP-Based Network—Case Study on The Indonesia Higher Education Network," *J. Commun.*, vol. 13, no. 10, pp. 612–617, 2018, doi: 10.12720/jcm.13.10.612-617.

[8] S. Jebur Taher, O. Ghazali, and S. Hassan, "A Review on Cache Replacement Strategies in Named Data Network," *J. Telecommun. Electron. Comput. Eng.*, vol. 10, no. 2–4, pp. 53–57, 2018.

[9] H. Situmorang, N. R. Syambas, T. Juhana, and I. Y. Matheus

Edward, “A Simulation of Cache Replacement Strategy on Named Data Network,” *Proceeding 2018 12th Int. Conf. Telecommun. Syst. Serv. Appl. TSSA 2018*, pp. 1–4, 2018, doi: 10.1109/TSSA.2018.8708796.

[10] J. hua Ran, N. Lv, D. Zhang, Y. yuan Ma, and Z. yong Xie, “On Performance of Cache Policies in Named Data Networking,” no. Icacsei, pp. 668–671, 2013, doi: 10.2991/icacsei.2013.160.

[11] Tanwir, G. Hendrantoro, and A. Affandi, “Combination of FIFO-LRU Cache Replacement Algorithms on Proxy Server to Improve Speed of Response to Object Requests From Clients,” *ARPN J. Eng. Appl. Sci.*, vol. 12, no. 3, pp. 710–715, 2017.

[12] H. Khelifi, S. Luo, B. Nour, and H. Moungla, “A QoS-Aware Cache Replacement Policy for Vehicular Named Data Networks,” *2019 IEEE Glob. Commun. Conf. GLOBECOM 2019 - Proc.*, 2019, doi: 10.1109/GLOBECOM38437.2019.9013461.

[13] “Mini-NDN.” https://github.com/named-data/mini-ndn (accessed Nov. 11, 2021).

[14] A. Gawande and L. Wang, “Need for Mini-NDN.” https://www.caida.org/workshops/ndn/1703/slides/ndn1703_agawande.pdf (accessed Nov. 11, 2021).

[15] B. Lantz, “Mininet: Rapid Prototyping for Software Defined Networks,” 2021. https://github.com/mininet/mininet (accessed Nov. 11, 2021).

[16] L. Afanasyev, Alexander; Mastorakis, Spyridon; Moiseenko, Ilya; Zhang, “Introduction — ndnSIM documentation.” http://ndnsim.net/current/intro.html (accessed Nov. 11, 2021).

[17] “GÉANT Topology Map,” 2018. https://www.geant.org/Networks/Pan-European_network/Documents/GEANT_Topology_Map_December_2018.pdf (accessed Nov. 09, 2021).