5G Network Simulator for Network Caching

Dongju Kim, Joonyoung Lim, and Younghwan Yoo
School of Computer Science and Engineering
Pusan National University
Busan, Korea
{rlaehdwn9097, ingyuh1015}@naver.com; ymomo@pusan.ac.kr

***Abstract*—This paper proposes a simulator for network caching research in a hierarchical 5G network. The proposed simulator adopts a modular structure to simply and quickly test new caching policies, and also includes well-known methods such as Leave Copy Everywhere (LCE) and Leave Copy Down (LCD) for performance comparison. The functions of combining existing caching policies and adjusting the number of components enable researchers to easily test their own caching policies. Additionally, the simulator provides the module to test a caching policy based on the Deep Q-Network (DQN). Recently, mobile edge caching (MEC), which stores frequently accessed content at the edge of the network, has been proposed to reduce the backhaul traffic. The proposed simulator also offers a way for evaluating MEC performance in hierarchical 5G wireless networks.**

***Keywords*—5G network, Network simulator, Mobile edge caching, Deep reinforcement learning**

I. Introduction

The rapid development of mobile communication technology and devices has enabled users to conveniently access a range of services via their mobile devices. Video content constitutes 70% of mobile network traffic, with the mobile network traffic reported at 108EB per month in 2022 Q3, and expected to increase to 453EB per month by the end of 2028, Where video content is expected to occupy 80% of traffic [1]. This increase in mobile traffic causes network backhaul traffic overload in the traditional hierarchical fifth generation (5G) network structure and also affects users' quality of service (QoS). In order to satisfy users' QoS, content providers such as Youtube or Netflix have placed content distribution networks (CDNs) around the world to solve this problem [2]. However, since a large portion of network traffic generated from users' content requests is occupied by duplicate requests for popular content, 5G wireless network congestion from user equipment (UE) to network backhaul is still expected to increase, affecting users' quality of experience (QoE) [3].

In order to cope with the increasing consumption of overlapping content in the hierarchical 5G network cloud structure, mobile edge caching (MEC) method has been proposed [4]. This method is storing commonly accessed content such as videos, music, and images at the edge of the mobile network, close to the user equipment (UE), in order to enhance the quality of service (QoS) and reduce latency. MEC also lessens the backhaul traffic in the network by caching popular content at the edge (Base station, UE), reducing the number of requests sent to the data center and reducing the amount of data that must be transmitted over the network's backhaul link. This leads to improved overall efficiency of the network and eases the strain on its resources. In addition, by using this method, it is possible to provide better services to users at a lower cost without modifying the overall physical network architecture to internet service providers (ISPs) and content providers [5].

To maximize the advantages of MEC, it is crucial to conduct experiments on the optimal caching policies. NS-3 [6], MATLAB [7], OMNeT++ [8], and OPNET [9] are currently popular simulators for 5G wireless networks, each providing a range of functions. However, configuring the environment for MEC experiments can be challenging, and requires a significant investment of time and skill to master the program and modify the simulator code.

Therefore, we propose a simulator that can perform MEC experiments in hierarchical 5G wireless networks.

- We provide an environment to experiment with content caching policies. Caching policy is divided into cache placement policy and cache replacement policy, and the basic policy is provided in the simulator. And it is provided as an application programming interface (API) so that users can freely modify the caching policy.
- It can be applied to different content types and scenarios for users. It provides sequential method and zipf distribution for scenario generation, so it can be provided according to the scenario type.
- The arrangement and number of Elements (UE, small cell, macro cell, data center) constituting the layers of 5G wireless can be configured freely.

The reminder of this paper is organized as follows: system model is presented in Section 2. In Section 3, we introduce overview of simulation process. Experiment with deep reinforcement learning is illustrated in Section 4 and the experiment results is shown in Section 5. Finally Section 6 presents our conclusions.

II. System model

This section introduces the overall architecture of our proposed simulator and explanations of major components. We also introduce components that are user configurable.

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C1095635).

A. Major Components

a) End Nodes/User Equipments: The UEs are the entity requesting the content. Content is requested from random UEs according to set scenarios. The request checks the content storage of the base station, and if there is the requested content in the content storage, the base station transmits the content to the UE. UEs are randomly generated in the set network environment area, and the number of UEs can be set as desired.

b) Base Stations: Base stations are divided into two different types of base stations: macro cell, and small cell. A macro cell is a traditional cellular base station that provides wide area coverage over a large geographic area. Macro cells are typically mounted on tall towers or rooftops and are designed to cover a radius of several kilometers. Small cells are low-power cellular base stations that are used to provide coverage and capacity in densely populated areas, such as urban centers, stadiums, and shopping malls. The arrangement and number of base stations are adjustable.

c) Data Center: A Data center has wired connections between the macro cell. It is located on the inner edge of 5g network and connects to the core network by combining user requests from base stations. It has larger content storage size than base stations. Also user can customize the storage size.

d) Core Network: The core network corresponds to network backhaul. Inside the core network are numerous routers and content providers. However, in this simulator, routers and different content providers are integrated into one core network. Therefore, it is assumed that the core network has all the content requested by the user and has a fixed content delivery delay.

e) Content Storage: Content storage is responsible for storing or deleting contents. It is located in the base stations, and the storage size of each tier is adjustable. In this storage system, heterogeneous content can be stored and deleted as complete content, rather than in individual chunk units.

f) Contents: Content consists of the types of images, videos, music, etc. that are requested by the user and provided by the content provider. For content, id and size are essential features, and the network simulator can port the content the user desires. Also, custom contents can have different sizes.

B. Generating Scenario

A Scenario is a sequential sequence of requested contents. There are two ways to generate scenarios. First, there is a method of generating in the order of scenarios specified by the simulator user. For example, when a simulator user simulates using time-stamped real data such as movieLens data [10], requests are generated sequentially according to the scenario index. Second, in the absence of a user-defined scenario, a

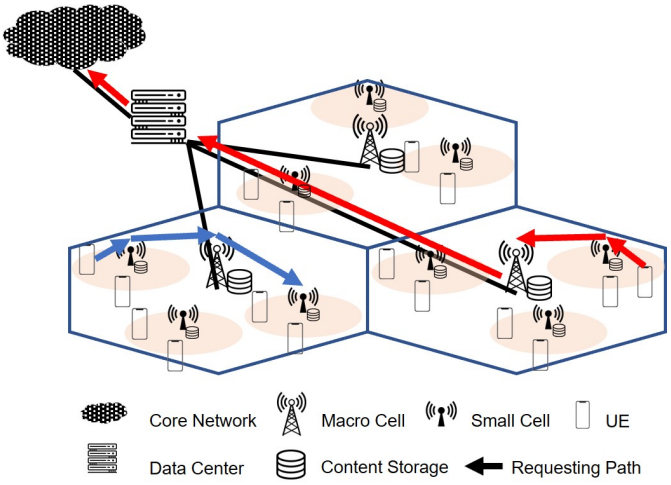

Fig. 1. Overall structure of network simulator.

scenario is generated using the Zipf distribution which is often used for determining content popularity by referring to the content's rank [11]. Assume that the contents are sorted by popularity, let total number of contents is M and the set of contents is $C = \{c_1, c_2, ..., c_M\}$. The set of content popularity is $P = \{p_1, p_2, ..., p_k, ..., p_M\}$ and let k be the rank of content. The popularity of k^{th} content can be expressed as

$$p_k = \frac{k^{-\beta}}{\sum_{m=1}^{M} m^{-\beta}} \tag{1}$$

where Zipf exponent $\beta \geq 0$ determines the skewness of popularity. When $\beta = 0$, popularity of contents is corresponding to a uniform distribution while higher β form a more skewed distribution. A scenario is generated based on the popularity of the content as determined by the Zipf distribution.

C. Caching Policies

Caching policies are an important part of the simulator. The aim of caching policy is to determine how and where content should be cached in the network to optimize content delivery and improve end-user experience. The provided caching policy is divided into placement policy and replacement policy.

a) Placement Policies: A placement policy refers to a strategy that determines the selection of a content storage to store the content. For example, the placement policy prioritizes serving content from the server closest to the user in order to minimize the time it takes for the content to reach the user. However, due to the limitations of storage capacity, it might be necessary to store content with high popularity in other content storage rather than in the closest content storage. Also the placement of the content within the hierarchical structure of the base station and data center may depend on whether the

content request is popular in a locally or widely. Therefore, the placement policy refers to the policy of choosing the appropriate content storage in the hierarchical structure of the 5G wireless network, considering the limitations of storage capacity and request preferences of users within the coverage of base station.

The simulator provides several types of placement policies as listed below. The placement policies include Leave Copy Everywhere [12], Leave Copy Down [13], and Random [14]. As placement policies are available as APIs, users can have the flexibility to implement their preferred policies.

- Leave Copy Everywhere (LCE): Under this policy, requested content is stored for every requested path's storage. With this policy, content exists in the entire requested path, thereby reducing the load on the original source and ensuring network stability.
- Leave Copy Down (LCD): Under this policy, requested content is stored in the BS just before the last BS or data center that received the request. With this policy, requirements of caching content reduce by limiting the number of storage.
- Random (RND): Under this policy, requested content is stored in a random storage located within requested paths. This policy store content without considering any of factors of network status.

b) Replacement Policies: The replacement policy refers to the policy that outlines the criteria for replacing the content when the content storage is full after the placement policy has been processed. For instance, when the content store reaches its capacity, it removes existing content according to the replacement policy in order to accommodate new content. By filling the free space with new content, it is expected to increase the cache hit rate and decrease the cache miss rate. The criteria for replacement can be based on various factors such as the frequency of access, the age of the content, and the size of the content. Therefore, the replacement policy refers to a strategy aimed at determining replacement criteria based on various content factors, to maximize the cache hit rate, and minimize the cache miss rate.

The simulator also provides several types of replacement policies, as listed below. The replacement policies include First In First Out [15], Least Recently Used [16], and Least Frequently Used [17]. Also replacement policies are available as APIs, users can have the flexibility to implement their preferred policies.

- First In First Out (FIFO): Under this policy, the content storage maintains a record of the time when the content is stored. When storage is full, it deletes the oldest content and makes room for new requested content.
- Least Recently Used (LRU): Under this policy, the storage maintains a record of the time when the content is accessed. When storage is full, it deletes the least recently accessed content and makes room for new requested content.
- Least Frequently Used (LFU): Under this policy, the BS storage maintains a record of the count of how many times content has been accessed. When storage is full, it deletes the least accessed content and makes room for new requested content.

D. Performance Metrics

The QoE-based performance metrics provided by the simulator to evaluate the performance of caching policies are as follows:

- Cache hit rate: This metric refers the number of times a user-requested content was served from the storage divided by the total number of user requests.
- Cache diversity: This metric refers the diversity of content by counting the number of different content within storage of the entire network [18].
- Cache redundancy: This metric refers the number of copies of the same content counted within storage of the entire network [19].
- Hop-count: This metric refers the total number of base stations within the requested path [20].
- Latency: This metric refers the entire delay of requesting and receiving requested content in requested path.

III. Overview of Simulation Process

Among the major components described in the previous chapter, we proceed with the setting of the base station. The coverage range, number, and size of content storage for each base station are set. Based on the settings, the base stations are randomly positioned within their coverage range. Data center are connected with marco cells and storage size is configured by setting. The UEs are randomly placed within the coverage of the designated number of base stations. Once the placement of the components in the network environment is completed, a prepared scenario is generated. The simulator user can choose one of two methods to generate a scenario, either by directly preparing a scenario or by using a zipf distribution. Before starting the simulation with the configured network environment and scenario, a caching policy must be specified. The user can select one of the provided placement policies, such as LCE, LCD, RND, and one of the provided replacement policies, such as FIFO, LRU, and LFU. The caching policy is provided through an API and the simulator user can set and use the desired placement policy and replacement policy. After configuring the major components, scenario, and caching policy, the number of rounds specified by the user is processed. Requests occur once per round, and they can be initiated by any random user. The request process checks the content storage of each component in the following order: small cell, macro cell, data center, and core network. Fig.1 shows two possible request paths for the requested content. The right request path shows the content being processed up to the core network when it is not found in the content storage of small cell, macro cell, and data center. The left request path

shows the content being processed by checking whether the requested content exists in the connected small cell and macro cell content storage. If the content is not found, the macro cell checks adjacent small cells for the requested content. If the requested content is found in the content storage of an adjacent small cell, the content is transferred from the small cell to which the UE is connected, through the higher-level macro cell. After the request path is determined, caching is carried out according to the caching policy set by the user, and QoE-based performance metrics are measured. Caching involves the selection of content storage in the base station and the addition and deletion of requested content according to the user policy. QoE-based performance metrics measure cache hit rate, hop count, and latency based on the request path. If content exists in the content storage before the core network, the cache hit count is updated for the cache hit rate. For hop count, the number of data centers and base stations in the request path is calculated and updated. Latency is calculated using the 3GPP TS 38.306 standard [21]. The formula for calculating latency can be expressed as follows:

$$Latency = d_{propagation} + d_{transmission} + d_{queueing} \quad (2)$$

where $d_{propagation}$ is propagation delay, $d_{transmission}$ is transmission delay and $d_{queueing}$ is queueing delay. Propagation delay can be expressed as follows:

$$d_{propagation} = \frac{dist_{m,n}}{l} \quad (3)$$

where $dist_{m,n}$ is physical distance between node m and node n. A node refers to a component in a network simulator environment. l is transmission medium speed. In this simulator, we assume l as light speed. Transmission delay and can be expressed as follows:

$$d_{transmission} = \frac{p_{size}}{D_{max}} \quad (4)$$

where p_{size} is packet size and D_{max} is the maximum throughput of 5G network. The formula of D_{max} can be expressed as follows based on 3GPP TS 38.306 standard:

$$D_{max} = 10^{-6} \sum_{j=1}^{J} \{V_l \cdot Q_l \cdot f \cdot R_m \cdot \frac{12 \cdot N_{PRB}^{BW,\mu}}{T_\mu} \cdot (1 - OH)\} \quad (5)$$

where V_l is maximum number of MIMO layers, Q_l is modulation order, f is scaling factor, R_m is Target code Rate, $N_{PRB}^{BW,\mu}$ is maximum number of physical resource block (PRB) for selected bandwidth (BW), frequency range (FR), allocated Sub carrier spacing (μ). T_μ is average OFDM symbol duration in a subframe and OH is overhead for control channels. In case of uplink and downlink, OH value is different. Queueing delay can be expressed as follows:

$$d_{queueing} = I \cdot (1 - I) \cdot \frac{s}{D_{max}}, \quad 0 \leq I \leq 1 \quad (6)$$

where I is traffic intensity. It is determined using a Gaussian distribution, with a value ranging between 0 and 1. Table I. shows configuration of the variables used in the calculation of latency. After calculating the latency, the simulator executes the next request until the round is over.

TABLE I
CONFIGURATION OF THE LATENCY VARIABLES

Variables	Values
V_l	4
Q_l	6
f	1
R_m	0.92578125
$N_{PRB}^{BW,\mu}$	133
BW	50Mhz
FR	1
μ	30kHz
T_μ	3.5714e-5
UpLink OH	0.08
DownLink OH	0.14

IV. EXPERIMENT WITH DEEP REINFOCEMENT LEARNING

This section introduces the application of our proposed network simulator and deep reinforcement learning (DRL) to network caching problems. In this experiment, the placement policy and replacement policy were learned by the DRL agent through their observations. The DRL agent used in this experiment is a Deep Q-Network (DQN) agent, which uses a technique that approximates a Q-table used in classic Q-learning to a deep neural network(DNN). Fig.2 shows simple structure of DQN agent.

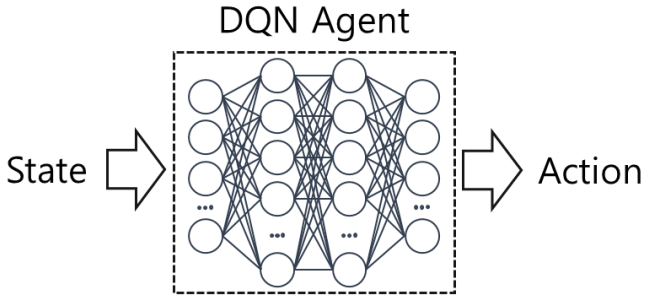

Fig. 2. Simple structure of DQN agent.

The DQN agent in our implementation consists of a set of fully connected layers, including an input layer, two hidden layers, and an output layer. The input layer consists of 105 neurons equal to the size of the state space of the network environment that the agent receives as input. The two hidden layers have 315 and 105 neurons respectively. In the hidden layers, the dropout rate is set to 0.2 to avoid overfitting. The output layer has 4 neurons equal to the number of the action

that agent can act on. Table II. shows hyper parameters of DQN agent used in experiment.

The state of the DQN agent consists of the content id and the number of accesses to the content in the request path. The action space of DQN agent consist of 4 actions. Each possible action set can be expressed as $A = \{a_0, a_1, a_2, a_3\}$. Actions a_0, a_1, a_2 are placement actions choosing storage among small cell, macro cell, and data center respectively. Action a_3 is the act of choosing none. The reward function of DQN agent consists of reduced latency and number of UEs by agent action.

We conducted experiments with 200 user equipment (UE), 36 small cells, nine macro cells, and one data center. The storage capacity of the small cells and macro cells is limited to 10 contents each, while the data center is designed to store up to 20 contents. Scenario of our experiment is based on viewers ratings on TV programs in korea. For training DQN agent, we ran 2,000 episodes, each consisting of 5,000 rounds. Under this configuration, our DQN agent training process is presented in Algorithm 1.

Algorithm 1 DQN Agent Training Algorithm

```
Initialize DQN Agent weights
for episode = 1 to E do                          ▷ E = Max Episode
    Initialize variables and reset network environment
    Get initial state of network
    for round = 1 to R do                        ▷ R = Max Round
        Execute network simulator round
        Get (Cache hit flag, State, Requested content,
        Path, Done) from round
        if Cache didn't hit then
            Get action from DQN agent
            Execute Step by agent action
            Get nextstate and reward
            Store transition (State, Action, Reward,
            Next state, Done) in replay buffer
            if buffer is fully stored then
                Decay epsilon
                Train DQN Agent
            end if
        else
            Update network environment
        end if
        Update target network
    end for
end for
```

V. Experiment Results

In order to compare QoE-based performance metrics, we conducted experiments between models that combined the provided caching policies and the DQN agent discussed in the previous section. Models were combined with LCE and LCD for the placement algorithm and FIFO and LFU for the replacement algorithm. The simulator settings used for these experiments were identical to those used in the previous section, with the exception of the round and episode. To improve the reliability of the results, we conducted 10 episodes of the simulation, each consisting of 50,000 rounds, and then calculated the average of the 10 episodes.

TABLE II
Configuration of the DQN agent variables

Hyper-parameters	Values
Optimizer	Adam
Loss function	MSE(Mean Squared Error)
Activation	ReLU
Learning rate	0.0001
Dropout rate	0.2
Epsilon decay	0.99995

Fig. 3 shows the average cache hit rate between DQN agent and composite models. The DQN agent achieves a cache hit rate of approximately 62%. When compared to the model with the lowest cache hit rate of 38%, which is achieved by the (LCE, FIFO) model, the DQN agent demonstrates improvement of 63.16%. Among the composite models, the (LCE, LFU) model delivers the best performance with a hit rate of around 50%, and the DQN agent performs 24% better than this model.

Fig. 4 shows the average hop count between DQN agent and composite models. The DQN agent achieves an average hop count of approximately 3.029428. In comparison, the (LCE, FIFO) model has the highest average hop count of 3.250857, indicating that the DQN agent demonstrates an improvement of 6.81%. Among the composite models, the (LCE, LFU) model exhibits the best performance, achieving an average hop count of approximately 2.867459. In comparison, the DQN agent has a hop count that is 5.35% higher than this model. The reason seems to be that the performance of the DQN agent is low in average hop count because the core network is also considered as one hop in our proposed network simulator.

Fig. 5 shows the average latency between DQN agent and composite models. The DQN agent achieves an average latency of approximately 0.00217. In comparison, the (LCE, FIFO) model, which has the longest mean latency of about 0.00337, demonstrates a latency that is 34.9% higher than that of the DQN agent. The best performing composite model is the (LCE, LFU) model, which achieves an average latency of about 0.00272. The DQN agent outperforms this model by 20.19%.

VI. Conclusions

In this paper, we proposed a hierarchical 5G network simulator for network caching that aims to prevent network backhaul overload and improve user QoS. Our simulator offers a straightforward approach to caching policy testing, with a simple code structure that facilitates easy code refactoring. This allows users a greater degree of flexibility in testing various caching policies. The simulator's modular design enables the adjustment of the arrangement and number of

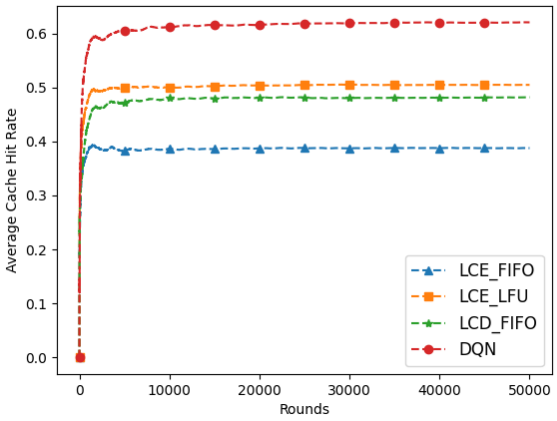

Fig. 3. Comparison of average cache hit rates between DQN agent and composite caching policy models.

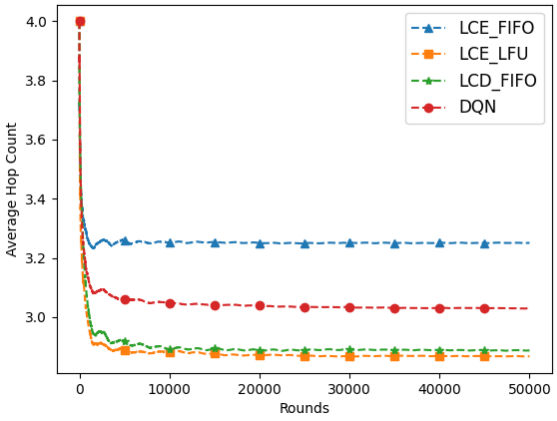

Fig. 4. Comparison of average hop count between DQN agent and composite caching policy models.

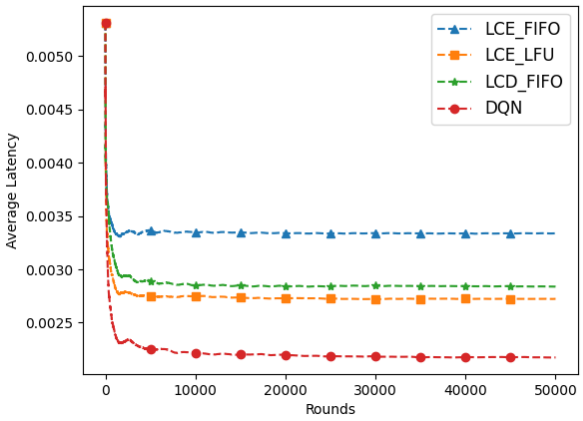

Fig. 5. Comparison of average latency between DQN agent and composite caching policy models.

components, and it includes a zipf distribution in scenario generation. By combining existing caching policies and setting them as control groups for desired tests, our simulator offers a platform for experimenting with optimal caching policies. As a result, we also experimented with the DQN agent to define the caching policy, which is an NP-hard problem, and confirmed that it outperformed the existing caching policy.

REFERENCES

[1] Ericsson, "Ericsson Mobility Report November 2022," 2022. [Online]. Available: https://www.ericsson.com/4a6f37/assets/local/mobility-report/documents/2022/ericsson-mobility-report-november-2022.pdf.

[2] K. Bouraqia, E. Sabir, M. Sadik and L. Ladid, "Quality of Experience for Streaming Services: Measurements, Challenges and Insights," in IEEE Access, vol. 8, pp. 13341-13361, 2020.

[3] X. Wang, M. Chen, T. Taleb, A. Ksentini and V. C. M. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5G systems," in IEEE Communications Magazine, vol. 52, no. 2, pp. 131-139, February 2014.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper, 2015.

[5] N. Golrezaei, A. F. Molisch, A. G. Dimakis and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," in IEEE Communications Magazine, vol. 51, no. 4, pp. 142-149, April 2013.

[6] The ns-3 network simulator. [Online]. Available: https://www.nsnam.org.

[7] The MATLAB 5G Tool Kit. [Online]. Available: https://www.mathworks.com/products/5g.html.

[8] The OMNeT++ discrete event simulator. [Online]. Available: https://omnetpp.org.

[9] The OPNET network simulator. [Online]. Available: https://www.riverbed.com/products/steelcentral/opnet.html

[10] MovieLens, "MovieLens 20M Dataset," [Online]. Available: https://grouplens.org/datasets/movielens/20m.

[11] L. Breslau, P. Cao, Li Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in Proc. IEEE INFOCOM. Conf. Comput. Commun. Proc. 18th Annu. Joint Conf. IEEE Comput. Commun. Societies. Future Now, pp. 126–134, 1999.

[12] Arif, Suki, Suhaidi Hassan, and Ibrahim Abdullahi, "Cache replacement positions in information-centric network.", The 4th International Conference on Internet Applications, Protocols and Services, pp. 54-58, 2014.

[13] N Laoutaris, H Che, and I Stavrakakis. The LCD interconnection of LRU caches and its analysis. Performance Evaluation, 63(7), 2006.

[14] W. Chai, D. He, I. Psaras, and G. Pavlou. Cache less for more in information-centric networks. pages 27–40, 2012.

[15] D. Rossi and G. Rossini, Caching Performance of Content Centric Networks Under Multi-Path Routing (and More), Relatório Técnico, Telecom ParisTech, Paris, France, pp. 1–6, 2011.

[16] M. Ahmed, S. Traverso, P. Giaccone, E. Leonardi, and S. Niccolini, "Analyzing the performance of LRU caches under non-stationary traffic patterns," 2013. [Online]. Available: arXiv:1301.4909.

[17] A. Jaleel, K. B. Theobald, S. C. Steely, Jr., and J. Emer, "High performance cache replacement using re-reference interval prediction (RRIP)," in Proc. ACM SIGARCH Comput. Archit. News, vol. 38, pp. 60–71, 2010.

[18] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," IEEE Commun. Surveys Tuts., vol. 17, no. 3, pp. 1473–1499, 3rd Quart., 2015.

[19] Z. Tong, Y. Xu, T. Yang, and B. Hu, "Quality-driven proactive caching of scalable videos over small cell networks," in Proc. IEEE 12th Int. Conf. Mobile Ad-Hoc Sensor Netw. (MSN), pp. 90–96, Dec. 2016.

[20] A. Ioannou and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," IEEE Commun. Surveys Tuts., vol. 18, no. 4, pp. 2847–2886, 4th Quart., 2016.

[21] 3GPP TS 38.306, v16.4.0, "Technical Specification Group Radio Access Network; NR User Equipment (UE) radio access capabilities,", 2020.