Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Learning-Based Adaptive Content Caching for Named Data Networking

Sushil Kumar Bagi

sushil.23183963@mygyanvihar.com

suresh gyan vihar university , jaipur https://orcid.org/0009-0009-1735-3404

neeraj kumar

suresh gyan vihar university https://orcid.org/0000-0002-4318-7704

Research Article

Keywords: Machine Learning, Q-Learning, Cache Replacement, ndnSIM, Named data networking

Posted Date: April 10th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-7765715/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Learning-Based Adaptive Content Caching for Named Data Networking

Sushil Kumar Bagi
dept. of Engg. & Technology
Suresh Gyan Vihar University
Jaipur, India
Author email :sushil.23183963@mygyanvihar.com

dep. of Engg. & Technology
Suresh Gyan Vihar University
Jaipur,India
neeraj.kumar1@mygyanvihar.com

Neeraj Kumar

***Abstract*: Efficient caching in Named Data Networking (NDN) is crucial for increasing content retrieval performance and reducing network congestion. Traditional caching strategies rely on heuristic-based content popularity estimation, which lacks adaptability to dynamic network conditions. In this paper, we propose a caching replacement technique, Q-CREPL (Q-learning-based cache replacement), which is a model-free machine learning approach using reinforcement learning to intelligently provide cache replacement decisions in NDN. Our method leverages real-time network interactions to dynamically adjust caching policies without requiring prior knowledge of content distribution. The Objective of this research paper is to develop a model-free cache replacement strategy that can dynamically learn and make decisions on the eviction of data items from memory in NDN. The methodology used to frame the cache replacement problem structure with Markov Decision Process and implemented with the help of the Q-Learning algorithm to make intelligent eviction decisions. An experimental result in the real environment of named data networking under two topologies that demonstrates that Q-CREPL consistently outperforms traditional strategies like Priority FIFO and LRU across all cache sizes. Its Cache hit ratio improves continuously as the cache size increases. However, in terms of latency in tree topology, LRU exhibit slightly lower latency at very small cache size due to its minimal decision overhead, but when cache size increases, Q-CREPL progressively outperforms both traditional strategies. It achieves latency levels during learning in state spaces that become comparable to LRU and Priority FIFO under varying memory size in real data traffic, and is simulated with the help of the ndnSIM simulator. The main contribution of this study is to enhance the performance of the future network, i.e., the NDN network, by using a cache replacement technique.**

Keywords— Machine Learning, Q-Learning, Cache Replacement, ndnSIM, Named data networking

I. INTRODUCTION

Named Data Networking (NDN) is a next future internet architecture that provides the option to the host-centric communication paradigm with a data-centric approach. In NDN, data item is retrieved by names other than IP addresses, and intermediate routers can cache data for fulfil the future requests, reducing retrieval delays and improving network efficiency. However, determining which content to cache and for how long remains in memory is a critical challenge. There are two types of cache replacement strategies are used (i) static cache replacement strategy (ii) dynamic cache replacement strategy. The example of static cache replacement strategies are FIFO, LRU, RR, LFU out of which LRU perform better [1], but they are based on fixed rule so we need to understand how machine learning can be useful when we have to take the decision based on multiple parameters. Traditional caching strategies estimate content popularity using predefined rules or historical access patterns. These methods struggle to adapt to evolving traffic dynamics and often fail to optimize cache utilization that's why we need to optimize caching decision. Machine Learning (ML) is a branch of artificial intelligence that relies on mathematical models to analyze data, recognize patterns, and generate predictions computationally. It can be viewed as a set of tasks in which a machine is able to make decisions based on the available data. The most critical step in any ML process is the training of a model, during which the system learns underlying patterns from data in order to enhance decision-making and prediction However, most existing studies in NDN applied Q-learning in areas such as forwarding or cache pollution mitigation, rather than directly for cache replacement with cache hit/miss rewards in named data networking. This gap motivates our work to apply machine learning algorithm in the caching decision in the area of NDN [2], [3]. The main contribution of this paper are as follows (i) we developed Q-Learning based cache replacement algorithm and compare the performance with the two traditional cache hit ratio i.e. LRU and Priority FIFO.(ii) we design the mathematical structure of problem with the help of markov decision process (MDP).(iii)compare the performance of Q-learning based cache replacement with two traditional cache replacement strategies. (iv) we evaluate the performance with the help of following metrics like cache hit ratio, latency and hop count. The proposed Q-Learning based cache replacement abbreviated as Q-CREPL, technique which used Q-learning algorithm to learn replacement decision and take the decision dynamically on data eviction. The new technique take the optimize caching decision based on real time data in NDN. It is reward based decision technique which guides the agent or program to take the decision based on environment. When there is a positive action taken by agent then this algorithm gives positive reward otherwise it gives negative reward as to give penalty for each wrong decision. The simulation results clearly indicate that the Q-CREPL technique consistently outperforms Priority FIFO and LRU strategies. It performs better to LRU and Priority FIFO strategies from varying content store size from 10 to 100. The rest of this paper is organised as follows. System overview in section 2. Related work regarding cache replacement in section 3. Overview of Q-learning in section 4.Implementation of proposed Q-learning based cache replacement strategy in section 5.Mathematical formulation of the Q-Learning-based caching model in section 6. Experimental setup in section 7. Performance Evaluation in

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

section 8.Result and discussion in section 9.Conclusion and Future research direction in section 10 and at last Bibliography & References in section 11.

II. SYSTEM OVERVIEW

The main architecture of Named data networking (NDN) basically has seven components.
1. Interest Packet
2. Data Packet
3. Consumer
4. Producer
5. Pending Interest Table (PIT)
6. Forwarding Information Base (FIB)
7. CS (Content store)

Where an interest packet is a request made by set of strings in which specific data content is mention. Data packet is made of name of content and actual data payload with content signature that is going too delivered. Consumer is a node who generates the request for content. Producer is a node who produces the content payload. The three data structure named PIT, FIB and CS are maintain by each router where PIT is table which maintains pending request of consumer. FIB is table which provides the information of forwarding strategy of particular interest or the request to next hop. CS is a memory where data is stored or cached for fulfilling the request locally without forwarding the request to the original data producer and it is also known as cache memory.
As in fig. (1), shows the basic architecture of NDN where the consumer send its request for data to producer with the help of routers. In which any node is based on principles of generating request, receiving request, sending response and forwarding the request. In this phenomenon only Interest packet is used by consumer to request a particular data contents from the network once network found the appropriate producer it will send the requested data packet to the particular consumer. This shift in architecture allows for more efficient data retrieval and reduces the reliance on traditional IP-based routing methods, ultimately improving overall network performance and user experience.
PIT is the table maintain by router which are responsible for maintaining the metadata of unfulfilled request of interest packet as well as number of faces from which interest packet generates and the possible output faces to which the same interest packet forward to obtain the data. This architecture not only enhance the network performance but also maintain the scalability of the network[4].

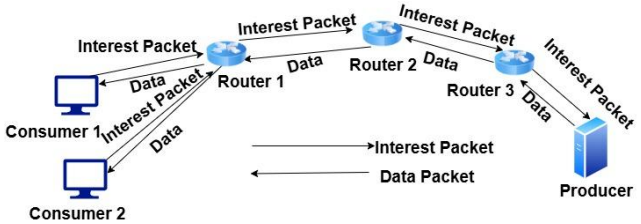


Fig.1. Basic NDN Architecture

III. RELATED WORKS

The primary goal of caching is to reduce the latency of content retrieval by caching the frequently need data nearest to consumers. The main point is how to handle the content store whenever the memory becomes full, at this point of time, replace the particular content from new arrival content with the help of cache replacement strategy [5][7].In Named data networking (NDN) caching is more crucial approach to deliver the content without connecting to the producer for every same request[6].Traditional replacement strategies such as First-In-First-Out (FIFO), Least Recently Used (LRU), and Least Frequently Used (LFU) are widely accepted due to their simplicity and low computational algorithm [7], [8].Out of these LRU policy most commonly used because of its adaptability to dynamic request pattern[9].However such static policies are limited capability to handle the dynamic and diverse traffic patterns. To address this lacking and limitations research has increasingly shifted towards intelligent and adaptive caching strategies. There so many traditional approach of cache replacement strategies are available like RR (Random Replacement): Randomly selects a cached content for eviction when the cache is full. Least Recently Used (LRU) : Evicts the Least recently accessed content. Least Frequently Used (LFU) : Evicts the least frequently item from memory. First in first out (FIFO) : The FIFO is used the concept that evict those data content from memory which one was oldest it means data items that was uploaded first in memory will be replaced by the new incoming data[10].Priority FIFO[11] is a cache replacement in which contents are grouped into priority classes and eviction is based on lowest priority class and each class maintain FIFO order.
The main limitation of above approaches is that they are static which are not capable to handle the adaptable caching decision in named data networking. So we designed the new caching decision based on machine learning where the decision could be based on learning.
There are so many research papers has been published in which machine learning played the vital role regarding enhance the decision making process for the improvement of caching strategy like in the paper [12] the author used graph neural network approach for caching and shown that GNN based caching has higher Byte hit ratio (BHR), Cache Hit Ratio (CHR) and lowest Average Latency Time(ALT).In the paper [13] author proposed The Apriori algorithm based caching which is used to forecast the upcoming data to be requested in near future and accordingly it maintain the cache. In the paper [14] author introduces a novel Q-learning based forwarding strategy, named QRF (Q-learning based Forwarding), for Named Data Networking (NDN) where in congested environments, QRF demonstrated superior performance in terms of Delay compared to Best Route which helps discover alternative, less congested paths and minimizes the load on better paths.

IV. OVERVIEW OF Q-LEARNING

Machine Learning (ML) models and algorithms can be broadly classified into three categories, depending on the nature of the available data and the approach used to interpret it: (i) Supervised Learning, (ii) Unsupervised

Learning, and (iii) Reinforcement Learning. Where supervised Learning is a paradigm in which models are trained on labelled datasets and both input data and corresponding outputs are provided. It educate the system based on input and outcomes mapping and use the training to prepare the model [15].Unsupervised learning means train the machine without knowing output. It searches the pattern in data and finds the cluster and classification for the given data set. Where Reinforcement learning is different paradigm in which agent is a program which interacts to the environment and makes the decision based on feedback of action's result. Q-learning is one of the model free reinforcement learning[15].

Q-learning also known as model free learning is one of the most applied representatives of reinforcement learning that works by learning on action-value function and it was developed by Watkins. This algorithm works by calculating the Q-value of state-action pairs. The main elements of Q learning are environment state, agent's action and reward. In the algorithm at each action, state and reward is used to calculate the Q-value by the Bellman equation[16]. It is also known off policy strategies. This important quality makes the Q-learning useful in applications such as robotics, game playing, computer networking and transportation field. In the Q learning the agents take complex decision-making processes to achieve desired outcomes. In transportation simulation, Q-learning can effectively model daily activity planning, allowing agents to optimize their activity sequences for maximum utility over time. This approach not only enhances the efficiency of transportation systems but also contributes to better urban planning and resource allocation. Recently with the general advances of machine learning, more variants of Q-learning like Deep Q-learning which combines basic Q learning with deep neural networks have been discovered and applied extensively [17].

It has two separate policies. One of them to participate in the optimization process and the other to explore the environment. As we know that policies are the functions that return either an action or the probabilities associated with each action.

These policies are given below in terms of Q- learning

(i) Target Policy

(II) Exploratory policy

4.1 Target Policy: The policy that participates in the optimization process is called Target policy and it is also called greedy policy because it will always select the action with the highest estimated Q value. It is denoted by π.Target policy chooses the action in a greedy manner. It selects the action with the highest estimated Q value. It can be considered that target policy is one that will optimize through the learning process.

i.e. $\prod(s) = \arg\ max_a Q\,(s,a)$ (1)

4.2 Exploratory Policy: It uses to face the environment and collect experience samples. It is denoted by b or b(s).Now our exploratory policy will be stochastic which means that every action will have a certain probability of being chosen. The agent will choose the actions that it will execute in the environment using the exploratory policy[18].Q-Learning update rule was based on below bellman equation learning:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma . Q(S_{t+1}) - Q(S_t, A_t)] \quad (2)$$

Where $Q(S_t, A_t)$ is Q value of each state, action pair and update $Q(S_t, A_t)$ based on the best available actions. Where **α** is learning rate and γ is the discount factor[18].

Reinforcement learning, particularly Q-learning, provides a more flexible approach by allowing the system to learn optimal caching strategies through continuous interactions with the network. The flow chart of Q learning algorithm shown in figure 2.

V. Implementation of Proposed Model-Free Q-Learning-Based Caching Strategy

5.1 System Model

We consider an NDN network where routers can cache contents and make caching decisions based on observed request patterns. The caching decision is framed as a reinforcement learning problem, where each router acts as an agent interacting with the network environment. There are various approaches and techniques to implement Q-learning. Here, we use a basic structure that gives a positive reward for each cache hit and a negative reward for each cache miss, and make intelligent caching decisions. The main components of our strategy are a) State representation, b) Action space, c) Designing of Q-Table, d) Reward Function, and e) Learning process. All components make up the Q-Learning Framework.

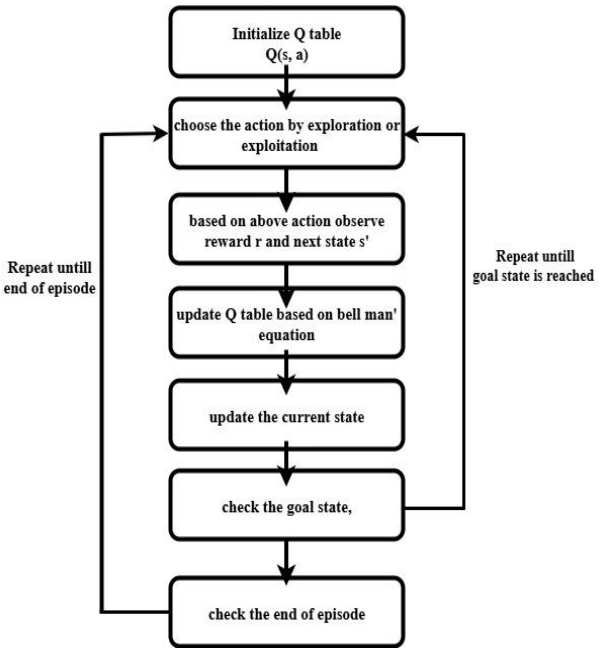


Fig. 2. Flow chart of Q learning Algorithm

5.2 Example of Q-learning with respect to caching:

Let's assume a simple scenario to understand how the Q table works. Suppose that the Q-table at the middle state with two sizes of cache memory as mention in table 1. Where data in current state of cache memory is (2, 4) and new data 1 comes then it will be updated the q-table as per table 2 with below calculation. Request 1 arrives while the

current cache state is (2,4) (so it's a miss).To understand purpose only we took positive reward +1 for each good action and negative reward -1 for each bad action. Then we must choose an eviction (evict 2 → next state = (1,4), or evict 4 → next state = (1,2)). Now here is the interesting part that picks the next state with a greedy approach, whose next state is the highest max Q:maxQ((1,4)) = max(0, 0) = 0, maxQ((1,2)) = max(0.975, −0.500) = 0.975 → higher, so evict 4 and move to (1,2).Now update Q table by equation 1,only Q((2,4), Evict) with α = 0.5, γ = 0.9, r = −1(eviction is bad action):

$Q_{new}((2,4),E) = Q(2,4)+0.5\ (-1+.9\times0.975 - 0)$

$Q_{new}((2,4),E) = 0.0+0.5(-1+.8775-0)$

$Q_{new}((2,4),E) = 0.0+-0.5\ (-0.1225)$

$Q_{new}((2,4),E) = -\ 0.06125$

Table 1: Intermediate Q-value Table

State (Cache Contents)	Action : Not Evict	Action: Evict
(1,2)	0.975	-0.5
(1,3)	0	0
(1,4)	0	0
(2,3)	0	-0.5
(2,4)	0.5	0
(3,4)	0	-0.5

Table 2: updated Q-value Table after request 1 (second Occurrence)

State (Cache Contents)	Action : Not Evict	Action: Evict
(1,2)	0.975	-0.5
(1,3)	0	0
(1,4)	0	0
(2,3)	0	-0.5
(2,4)	0.5	-0.06125
(3,4)	0	-0.5

Since Q-learning is a continuous process, it employs greedy approach to make decisions.

5.3: Complexity analysis of Algorithms:

The time complexity and space complexity are shown in below table 3

Table 3: Complexity of among strategies

Complexity	Priority FIFO	LRU	Q-CREPL
Time Complexity	O(1)	O(1)	$O(N^2)$
Space Complexity	O(N)	O(N)	O(N)

Q-CREPL does full CS scan per eviction after outer while loop to check the condition of cache size and limit.

the Q-learning-based caching takes $O(N^2)$ time, it gives a higher cache hit ratio (CHR) at the cost of higher computation but the practical cost remains bounded due limited cache size and periodic execution of algorithm.

VI. MTHEMATICAL FORMULATION OF THE Q-LEARNING-BASED CACHING MODEL

So accordingly the mathematical representation of Q-Learning-Based Cache Replacement design below

1. Problem Modelling

The proposed cache replacement problem represented as a Markov Decision Process (MDP) define by four touple.

$M=(S,A,R,\gamma)$

where S is the state space,A is the action space , R is the reward function and γ is the discount factor.

2.State Space: Each State is corresponds to individual data item of content store(CS).In the implementation hasing is used to denote the state.

$S=h(n)$

3. Action Space:

For each state $s \epsilon S$, the program or agent take the following actions

$A=\{a_0,a_1\}$

where

a_0=Evict

a_1=Keep

4.Reward Function

The Reward function we decide based on the observation during run time.

r(s,a)={+5 if cache hit occurs (when content is refresh)
-3 if cache content is not reused (miss)
-5 if eviction is triggered due to cache overflow
0 otherwise}

This reward make the agent to take the correct decision about steps in cache replacement environment.

5. Q-Learning Update Rule

The Q-values are updated using the standard Bellman Optimality equation:

$$Q(s_t,a_t)\leftarrow Q(s_t,a_t)+\alpha[r_t+\gamma.\max a` \epsilon A Q(s_{t+1},a`)-Q(s_t,a_t] \quad (3)$$

where $\alpha\epsilon[0,1]$ is the learning rate,

$\gamma\epsilon[0,1]$ is the discount factor

r_t is the reward observed after executing action a_t

6. Eviction Decision Policy:

The eviction decision is made by selecting the content with the highest Q-value for the eviction action in cache replacement.

$S^* = \arg\max s\epsilon C\ Q(s,Evict)$

where C is the data item of content store and selected content S* is evicted from content store.

VII. EXPERIMENTAL SETUP

The new cache replacement technique Q-CREPL was simulated in the environment of named data networking with the ndnSIM simulator. In this experiment, we used two topologies for the experiment: (i)Tree Topology, (ii) Linear

Topology, where in the tree-like topology, consumers are installed on leaf nodes, and producers are at the root of the tree. Each consumer requests data from the same namespace: /root/1, /root/2.As shown in Figure 3, the consumers send the interest packets to the producer, and the producer replies with data packets to the consumers. The router one and router two, as intermediate nodes, can cache the data, and if the same request comes from consumers, then it will reply from the routers themselves rather than the producer. In the Linear Topology, all nodes of consumers, routers and producers are connected in a linear fashion. Same as depicted in fig. 4, in which consumers are connected to router one, the first router is further connected to the second router, and the second router is connected to the producers.

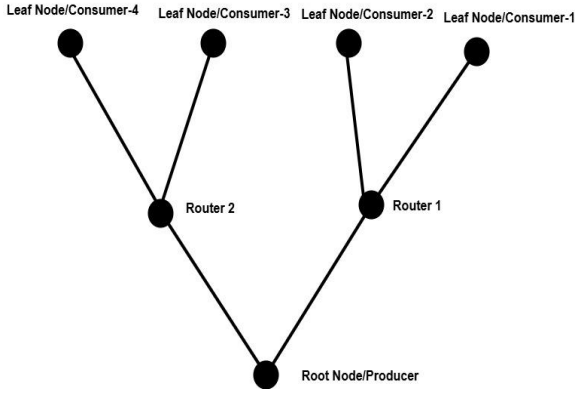


Fig. 3: Tree Topology

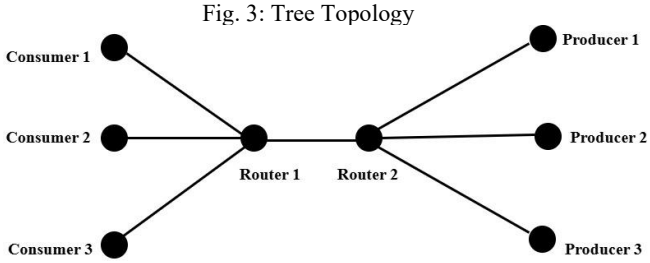


Fig.4: Linear Topology

We evaluate our Q-learning-based caching strategy using a custom NDN simulation environment implemented in c++ files. In this experiment, the topology has been designed in such way so that the Linear Topology provides a more diverse environment because it uses multiple name spaces, unlike the Tree topology.

VIII. Performance Evaluation

For the performance evaluation of among Q-CREPL, LRU and Priority FIFO, simulation is done with ndnSIM simulator and the simulation parameters are shown in table 4.The performance evaluation can varies from high or low depending on the below simulation parameters. In the both topologies we passed with high data constraints in order to check the robustness of algorithm.

TABLE 4: SIMULATION PARAMETERS

1. Tree Topology 2. Linear Topology	
Parameter	**Value**
Request Rate	100 request per sec
CS size	10/20/30/40/50/60/70 /80/90/100 (Entries)
Cache Strategy	Q-CREPL ,Priority-FIFO & LRU
Forwarding Strategy	Best Route
Simulation Time	100 s
Number of topology nodes	7 (Tree Topology)
	8 (Linear Topology)
Number of contents	500
q-Plateau Parameter	0.8
s- Skew Parameter	1.2
Learning rate	0.5
Discount Factor	0.9

IX. Results and Discussion

The results obtained from the simulation are presented in the table 5, 6 below.

Table 5. Calculated Results regarding Average CHR for Priority FIFO ,LRU and Q-CREPL strategy for both Tree and Linear Topology

1.	**Tree Topology**		
CS-Size	Priority FIFO	LRU	Q-CREPL
10	26.039%	30.564%	46.574%
20	39.092%	45.501%	58.341%
30	46.824%	53.851%	63.525%
40	52.581%	59.149%	66.908%
50	56.858%	63.287%	69.305%
60	60.208%	66.568%	71.093%
70	63.425%	69.405%	72.724%
80	65.850%	71.630%	73.643%
90	67.874%	73.356%	74.770%
100	69.662%	74.921%	75.466%
2.	**Linear Topology**		
CS-Size	Priority FIFO	LRU	Q-CREPL
10	2.731%	2.969%	17.900%
20	7.354%	9.121%	27.299%
30	11.239%	14.613%	33.326%
40	14.546%	19.517%	37.321%
50	17.368%	23.360%	40.185%
60	19.800%	26.765%	42.447%
70	22.315%	29.786%	44.310%
80	24.561%	32.334%	45.976%
90	26.567%	34.548%	47.203%
100	28.304%	36.575%	48.687%

Table 6. Calculated Results regarding Average Latency among Priority FIFO ,LRU and Q-CREPL strategy.

1.	**Tree Topology**		
CS-Size	Priority FIFO	LRU	Q-CREPL
10	5233.951 us	5086.867 us	5202.882 us
20	4795.263 us	4582.589 us	4098.111 us
30	4530.04 us	4302.721 us	3904.781 us
40	4340.828 us	4121.497 us	3791.168 us
50	4201.584 us	3986.994 us	3712.324 us
60	4087.214 us	3876.588 us	3649.979 us
70	3989.462 us	3790.565 us	3595.858 us

80	3908.019 us	3720.07 us	3555.778 us
90	3840.642 us	3660.85 us	3516.413 us
100	3780.271 us	3608.779 us	3491.823 us
2.	**Linear Topology**		
CS-Size	Priority FIFO	LRU	Q-CREPL
10	58361.451 us	58232.811 us	53437.768 us
20	55117.748 us	54117.493 us	47008.239 us
30	52609.957 us	50779.448 us	42891.586 us
40	50595.16 us	48110.516 us	39873.047 us
50	48899.33 us	46036.236 us	37821.586 us
60	47487.941 us	44298.498 us	36790.277 us
70	46168.477 us	42870.352 us	35857.638 us
80	44994.213 us	41673.687 us	35133.013 us
90	44046.395 us	40608.325 us	34521.705 us
100	43149.845 us	39705.171 us	33891.325 us

9.1 Hop Count: The hop count is calculated for Priority FIFO, LRU, and Q-CREPL using the file cs-tracse.txt, generated by the ndnSIM simulator. After analysis by a Python program, the results are shown in Table 7, which clearly shows that the lowest hop count is obtained with the Q-CREPL strategy. This technique enhances the performance of named data networking by providing the data to the nearest routers 1 and 2 to the consumer.

Table 7: Statistical Comparison of Hop Counts Across Caching Strategies

Topology	Priority FIFO	LRU	Q-CREPL
Tree topology	1.305	1.274	1.234
Linear Topology	1.873	1.829	1.688

9.2 Statistical analysis:

The two-tailed test is conducted, and the t-value and p-value are obtained (see Table 8). They are compared with alpha (α), and the significance of the difference between the algorithms is determined.

Table 8: Statistical comparison of cache replacement techniques based on t-values and p-values.

Topology	**Comparison**	**t-value**	**p-value**	**Significance**
tree	Q-CREPL vs LRU	3.94	0.0034	Significant
tree	Q-CREPL vs Priority FIFO	7.55	3.49e-05	Highly Significant
Linear	Q-CREPL vs LRU	21.04	5.81e-09	Extremely Significant
linear	Q-CREPL vs Priority FIFO	29.00	3.36e-10	Extremely Significant

As per table 5 and 6 under tree topology we can say that Q-CREPL strategy perform higher CHR than LRU and Priority- FIFO for content store size up to 100.This is expected as LRU and Priority FIFO performs on deterministic static technique without learning the network adaptability. At cs size 100 the average cache hit ratio of Q-CREPL still slightly higher with the value 75.466 % in comparison to 74.921% (LRU) and 69.662 % (Priority FIFO) also depicted in fig. 5 where at cs-100 green line is slightly above with black line.

Under Linear topology also Q-CREPL cache hit ratio is always high incomparision to LRU and Priority FIFO from content store 10 to 100.Same is depicted in fig. 7.

Conversely, under tree topology, In terms of Latency performance of all three strategies are evaluated with varying content size from 10 to 100.At small content store like cs-10 the LRU demonstrates the lowest average latency due to its underlying mechanism and their decision making system. From cs-20 to cs-100 the average latency of Q-CREPL lower with respect to others two methods.Same is depicted in fig. 6. This Latency results confirms that adaptive cache replacement outperform static heuristic strategy at larger networks. It is also noticed that all time the priority FIFO performance down in terms of cache hit ratio and latency with respect to LRU and Q-CREPL strategies. where under linear topology Q-CREPL latency is lowest among all three strategies. Same it is seen in fig. 8.It is further also noticed that overall cache hit ratio under linear topology is less for all strategies, it is just because we designed and created the linear topology where each consumer and producer operated within a mutually exclusive namespace in order to verify the robustness of algorithms. Where in Tree topology we designed simple scenario where each consumers send the request with single namespace and all producers sends response with same namespace that's it same request high overlap manner resultant high cache hit ratio overall.

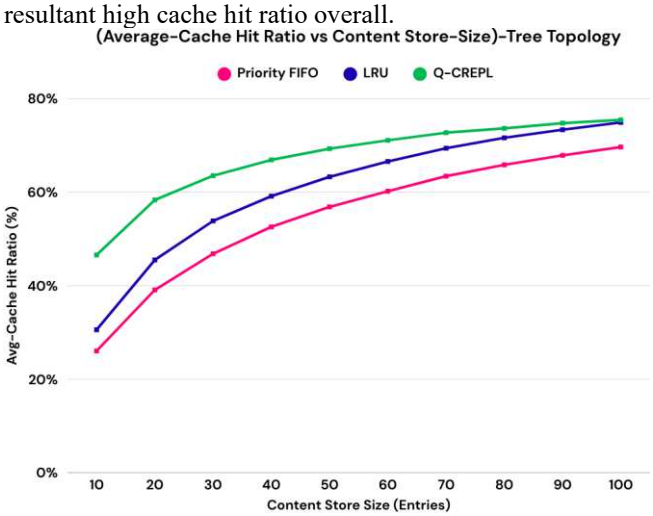


Fig.5: Avg. Cache Hit Ratio vs. content size for Priority FIFO, LRU and Q-CREPL under Tree Topology

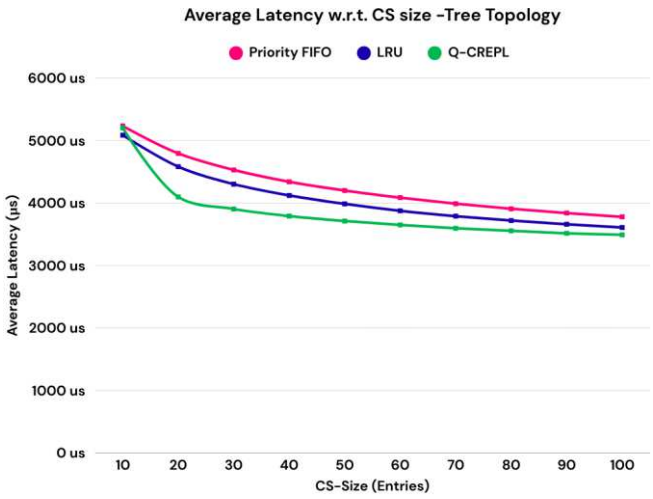


Fig. 6. Avg. Latency vs. CS-Size for Priority FIFO, LRU and Q-CREPL under Tree Topology

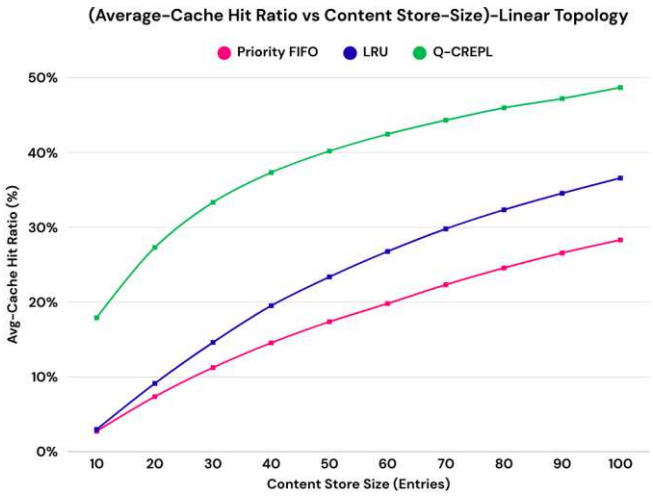


Fig. 7. Avg. Cache Hit Ratio vs. content size for Priority FIFO, LRU and Q-CREPL under Linear Topology

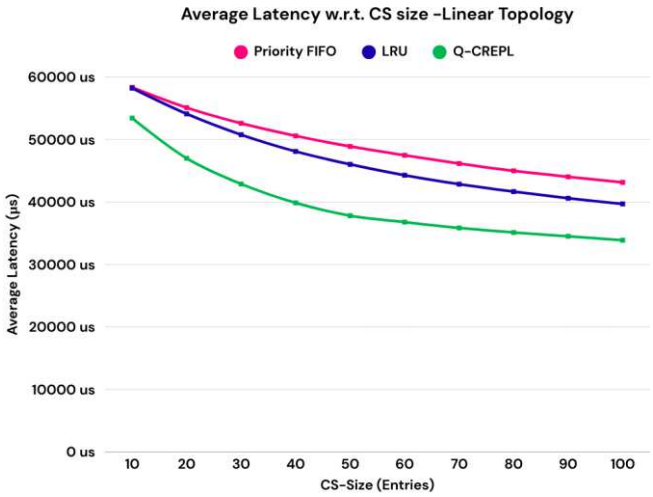


Fig. 8: Avg. Latency vs. CS-Size for Priority FIFO, LRU and Q-CREPL under Linear Topology

X. Conclusion and Future Work

Named data networking is a futuristic network concept in which data retrieval is based on data names rather than on location. Caching is a significant feature of NDN, in which the most recently requested data is stored at the nearest router. Hence, it is available for the same request without reconnecting to the producer. The cache replacement algorithm plays a crucial role in replacing data items from memory when it becomes full. Machine learning plays a vital role in training the machine to make decisions using the learning concept, with reinforcement learning as the approach. Q-learning is one of the off-policy machine learning algorithms used here with a cache replacement technique. Based on various parameters, including the results of ndnSIM simulations on cache hit ratio, latency, hop count, and complexity, and a statistical two-tailed test, we can say that machine learning based cache replacement technique obtain high cache hit ratio and low latency to traditional strategies (i.e., priority FIFO and LRU) with high computational cost. where traditional cache replacement algorithms are static and cannot adapt to dynamic network conditions. Future research should develop a cache replacement algorithm using machine learning, leveraging PIT and FIB tables, with low computational cost to improve the performance of named data networking.

XI. Bibliography & References

[1] M. I. Zulfa, A. Fadli, A. E. Permanasari, and W. A. Ahmed, "Performance comparison of cache replacement algorithms onvarious internet traffic," J.INFOTEL, vol. 15, no. 1, pp. 1–7, Feb. 2023, doi: 10.20895/infotel.v15i1.872.

[2] A. Hidouri, H. Touati, M. Hadded, N. Hajlaoui, P. Muhlethaler, and S. Bouzefrane, "Q-ICAN: A Q-learning based cache pollution attack mitigation approach for named data networking," *Computer Networks*, vol. 235, p. 109998, Nov. 2023, doi: 10.1016/j.comnet.2023.109998.

[3] Y. A. B. L. de Sena, K. L. Dias, and C. Zanchettin, "DQN-AF: Deep Q-Network based Adaptive Forwarding Strategy for Named Data Networking," 2020, *arXiv*. doi: 10.48550/ARXIV.2010.09897.

[4] L. Zhang *et al.*, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, July 2014, doi: 10.1145/2656877.2656887.

[5] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*, Helsinki Finland: ACM, Aug. 2012, pp. 55–60. doi: 10.1145/2342488.2342501.

[6] L. Zhang *et al.*, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, July 2014, doi: 10.1145/2656877.2656887.

[7] Donghee Lee *et al.*, "LRFU: a spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, no. 12, pp. 1352–1361, Dec. 2001, doi: 10.1109/TC.2001.970573.

[8] S. Podlipnig and L. Böszörmenyi, "A survey of Web cache replacement strategies," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 374–398, Dec. 2003, doi: 10.1145/954339.954341.

[9] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed Cache Management in Information-Centric Networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 10, no. 3, pp. 286–299, Sept. 2013, doi: 10.1109/TNSM.2013.052113.120382.

[10] L. Saino, I. Psaras, and G. Pavlou, "Icarus: a Caching Simulator for Information Centric Networking (ICN)," in *Proceedings of the Seventh International Conference on Simulation Tools and Techniques*, Lisbon, Portugal: ICST, 2014. doi: 10.4108/icst.simutools.2014.254630.

[11] E. T. D. Silva, J. M. H. D. Macedo, and A. L. D. Costa, "NDN Content Store and Caching Policies: Performance Evaluation," *Computers*, vol. 11, no. 3, p. 37, Mar. 2022, doi: 10.3390/computers11030037.

[12] J. Hou, H. Xia, H. Lu, and A. Nayak, "A Graph Neural Network Approach for Caching Performance Optimization in NDN Networks," *IEEE Access*, vol. 10, pp. 112657–112668, 2022, doi: 10.1109/ACCESS.2022.3217236.

[13] C. A. Kerrache, G. Rathee, Y. Guellouma, H. Gasmi, and B. Ziani, "Towards an Efficient and Secure Cache Management Using Apriori-Based Interests Prediction in Named Data Networking," in *2023 11th International Conference on Intelligent Systems and Embedded Design (ISED)*, Dehradun, India: IEEE, Dec. 2023, pp. 1–6. doi: 10.1109/ISED59382.2023.10444543.

[14] Y. Mordjana, B. Djamaa, and M. R. Senouci, "A Q-learning based Forwarding Strategy for Named Data Networking," in *2021 International Conference on Networking and Advanced Systems (ICNAS)*, Annaba, Algeria: IEEE, Oct. 2021, pp. 1–6. doi: 10.1109/ICNAS53565.2021.9628982.

[15] A. F. A. H. Alnuaimi and T. H. K. Albaldawi, "An overview of machine learning classification techniques," *BIO Web Conf.*, vol. 97, p. 00133, 2024, doi: 10.1051/bioconf/20249700133.

[16] S Manju and M Punithavalli, "An analysis of Q-learning algorithms with strategies of reward function," *International Journal on Computer Science and Engineering*, vol. 3, no. 2, pp. 814–820, 2011.

[17] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019, doi: 10.1109/access.2019.2941229.

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, Second edition. in Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018.