الجـــــــمهورية الجـــزائرية الديمـــــــــــقراطية الشـــــــعبية

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التـــــــــــعليم العـــــالي و البـــــــــــحث العـــــــلمي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جـــــامعة عمـــــــــــار ثلـــــــيجي بالاغـــــــواط

UNIVERSITY OF AMAR TELIDJI – LAGHOUAT

كليـــــة العلـــوم

FACULTY OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

Master Thesis

Domain: Mathematics and computer science

Specialty: Computer science

Option: Networks, Systems and Distributed Applications

By:

Hocine GASMI

THEME

Development of a Machine learning-based in-network Caching strategy for NDN

Jury members

Dr. Younes Guellouma	President
Dr. Leila Benarous	Examiner
Dr. Chaker Abdelaziz Kerrache	Advisor

2020/2021

Dedication

Alhamdulillah the most merciful the most powerful, then I want to thank my parents, my colleagues, friends, and everyone who helped from near or far.

~ Hocine GASMI

Acknowledgment

Alhamdulillah, I want to give my thanks to my supervisor **Kerrache Chaker Abdelaziz** who guide me through this work and gave me precious advice and insights. I want also to thank my friend and colleague **Abdelkader Herouala** who guide me greatly in helping in the experiments of this work. And everyone else who supported me.

الملخص

ذاكرة التخزين المؤقتة تعد مكون مهم في اي شبكة, في نموذج المقترح من الـ NDN تعد ذاكرة التخزين المؤقتة عنصر مهم جدا و ضروري في الشبكة و تعتمد عليه قيمة الاداء للشبكة. اولا سوف نتحدث عن الفرق بين الانترنت الكلاسيكية التي تستعمل الـ IP و الـ NDN, و ايضا تعلم الالة **(**Machine Learning**)**, سوف نتطرق الى استخدام نوع من تعلم الالة على مستوى التخزين المؤقت في عملنا هذا. خوارزمية الـ Apriori من الخوارزميات الـ Supervised التي تتناسب مع عملنا و ذلك لتوفر المعطيات في مرحلة التعلم, و اخراج لنا قواعد الارتباط. تلك سوف تساعدنا على تحديد ما هي البيانات المطلوبة مستقبلا, ذلك سوف يزيد نسبة الاصابة في البحث على مستوى ذاكرة التخزين المؤقتة للحلقة داخل الشبكة. ذلك يعود على كل من المستعمل بنقص وقت الانتظار بعد ارسال الطلب, ضغط اقل على الخادم الموفر للبيانات المطلوبة. بعد التجريب تحصلنا على معيار على شكل نسبة الاصابات في ذاكرة التخزين المؤقتة و التي بدورها حددت اداء المنهجية المقدمة و التي تتكون في اقتراح طلب البيانات المحتمل طلبها مستقبلا بزيادة قدرا **3.2%** في حجمي ذاكرة التخزين المؤقتة, اما في حجم **80%** فنتيجة متقاربة جدا مع كلا البيانين يصلان الى نسبة **90%** اصابة.

كلمات مفتاحية: ذاكرة التخزين المؤقتة, تسيير ذاكرة التخزين المؤقتة. تعلم الالة. Apriori, شبكة البيانات المسمات (NDN)

Abstract

Cache management is an important component in any network, and more is that it's essential in the NDN architecture. In this work first, we will talk about the difference between IP (Internet Protocol), NDN (Named Data Network), the cache management, its types, and the importance of that in the NDN architecture. Including ML (Machine Learning), using it at the cache level is the focus of our work. The Apriori algorithm is supervised learning that is suited for our work as we have the data for this learning approach, after that we find the association rules, to recommend the next requested data. A low hit ratio leads to the increase of requesting content from the content provider, for the client that means more access latency and pressure on the server. The result obtained determines that the performance of the network and its influence on the cache increased by 3.2% in the two content store sizes of 20 and 40, the size of 80 shows an increase of the cache hit ratio for both curves as identical with both reach 90%.

Keywords: Cache, Cache management, Machine Learning, Apriori, Named Data Network.

Contents

Abbreviations

NDN: Named Data Network

IP: Internet Protocol

TCP: Transmission Control Protocol

CS: Content Store

FIB: Forwarding Information Base

PIT: Pending Interest Table

DNS: Domain Name System

TLD: Top-Level Domain

HTTP: Hypertext Transfer Protocol

SYN: synchronize

ACK: Acknowledgement

ICN: Information-centric networking

CCN: Content centric networking

ICMP: Internet Control Message Protocol

FIFO: First In First Out

LRU: Least Recently Used

LFU: Least Frequently Used

CCR: Cache Content Replacement

DONA: Data-Oriented Network Architecture

PSIRP: Publish-Subscribe Internet Routing Paradigm

RTT: Round Trip Time

LPM: Longest Prefix Match

RSA: Rivest–Shamir–Adleman

DSA: Digital Signature Algorithm

DoS: Denial-of-Service

DDoS: Distributed Denial-of-Service

List of Figures

Introduction

Today needs require much more discussion of other fields to make use of the advancing of the technology we have today.

The classical IP architecture stayed a good architecture for a long time, but the shift and change of today need requirement for another architecture to arise and offer a suited and better approach especially for content that it's the main component of today's interactions.

Caching become a real essence of any network, optimizing and management of it increase the performance in many ways and many different aspects, the classical management may not always give us the best performance, here, considering Machine Learning can offer us a great advantage in the decisions and management of cache, due to the flexibility of it versus the standard approaches. The different learning algorithms to choose from are many and broad, choosing is up to the role played by it and the need of it to solve a specific problem.

Besides, Data Mining (DM) is a subfield of artificial intelligence (AI), it is the science that by exploring and analyzing huge blocks of information, we could extract meaningful trends and patterns. Here, Machine Learning can offer a great advantage in decisions and management of cache, due to its flexibility. The different learning algorithms to choose from are many, and that is up to the role played by it.

The Apriori algorithm is supervised machine learning, can be suitable for some approaches of caching, like our approach for instance that we will use the frequent pattern to extract the association rules to apply on the cache of NDN for increasing the cache hit ratio.

This approach will discuss the datasets and the steps through achieving the result, the simulation environment, and the tools used for that.

This work is divided into three main chapters:

From IP to NDN: is the first chapter talking about the classical architecture and its features and drawbacks, the Named Data Network, and finally the Cache Management.

Machine Learning: This is the second chapter that talks about the different types of Machine Learning, some of the examples, and the Apriori Algorithm.

Apriori-Based Cache Management: This is the final chapter that talks about the approach and methods for increasing the performance of the Cache.

1 From IP to NDN

1.1 Introduction

The internet of the present is based on the TCP/IP protocol, the IP address is necessary to any node to establish connections or receive data from other nodes. To receive a certain data you need first the address of the provider (IP address) of that data, and the provider will send data to the IP address of the requester, making it host-centric; focusing on the provider of the data rather than the data itself, and that comes with cons and pros that we will discuss.

1.2 Internet Protocol

The IP (Internet protocol) is responsible for delivering the packets in a network based on the IP addresses in the packets from source to destination. We will see the features and the characteristic of it and then will talk about the drawbacks and the problems that it can't solve efficiently for our today needs.

1.2.1 Features

At the moment the industry is using TCP/IP protocol for the last five decades. It's a tested and robust protocol, we will list the features the protocol still has.

1.2.1.1 Physical Network Independence

The underlying network hardware is hidden thanks to the IP, any new physical network can work by implementing a new driver that connects to the internet underneath IP, and that allows TCP/IP to integrate to any network.

1.2.1.2 Interoperability

In every environment nowadays we have many devices working with compatibility, the TCP/IP gives us that by crossing all platform/multi-vendor barriers.

1.2.2 Overload / Drawback

In every single home we will found many and different types of devices all connected to the internet that makes a huge amount of traffic requests and replies, also make a lot of pressure on the servers, websites, applications, and services with the loss of energy by using a lot of resources. The other side that is affected is the client, when a server is down or is satisfying requests slowly due to the capacity limit of the network resources, the client will lose time, or s/he will never get the desire data.

1.2.2.1 Respond Time

Internet devices use IP addresses to communicates and reach other devices on the internet, every website has a unique IP address, humans cannot remember IP addresses. The solution for that is the Domain Name System (DNS) that translates domain names to IP addresses, Figure 1 shows describe it. When a user wants to access a website firstly the IP address of the website is retrieved, the steps of that is as follow:

- The user types the name of the website like ‘example.com’ the query is sent to the internet and received by the DNS server.
- The resolver DNS forwards the request of ‘example.com’ to the DNS root name server.
- The root server then responds to the resolver with the address of the Top Level Domain (TLD) DNS server, the request is pointed to the .com domains.
- The resolver now requests the .com TLD.
- The TLD server returns a response with the IP address associated with the domain name ‘example.com’.
- The DNS resolver responds to the user (web browser) with the IP address of the domain.
- The browser makes an HTTP request to the IP address.
- The server of the IP address returns the web page.

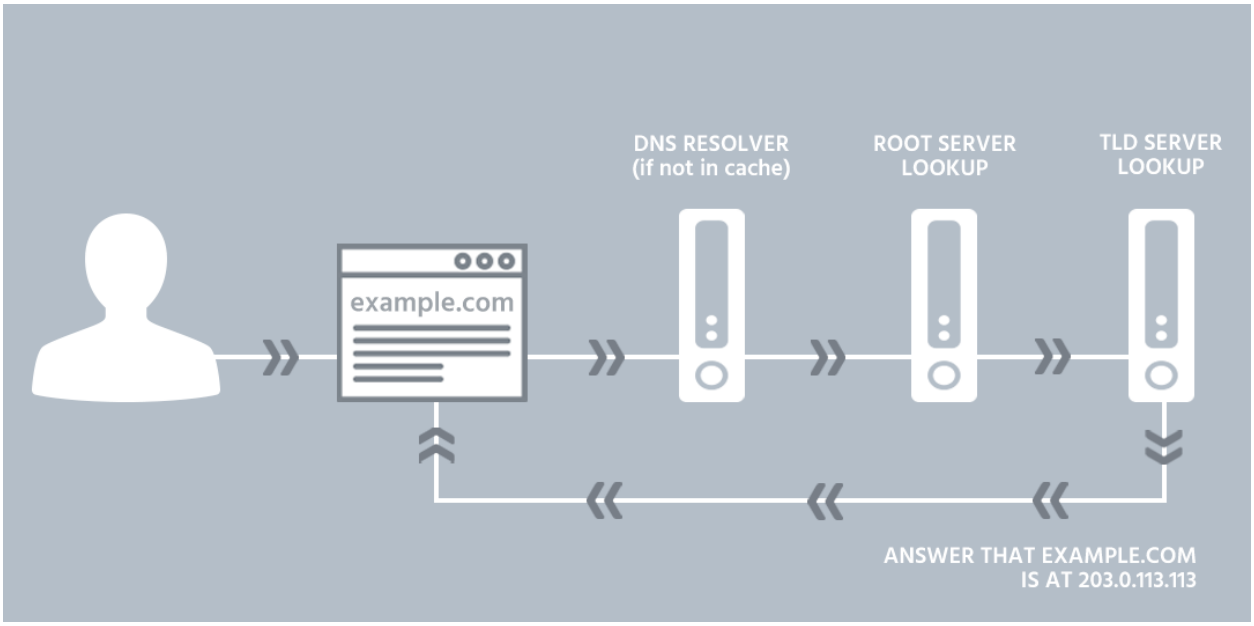

FIGURE 1: DNS REQUEST [1]

As we saw all those steps are done only for getting the IP address of the server we want to reach, which makes a time loss for the user not to mention the network pressure.

1.2.2.2 Establishing connections

Connection establishment in TCP is done by three-way handshake, Figure 2 explain the steps:

1. The client sends a SYN.
2. The server reply with SYN-ACK.
3. The client sends ACK.

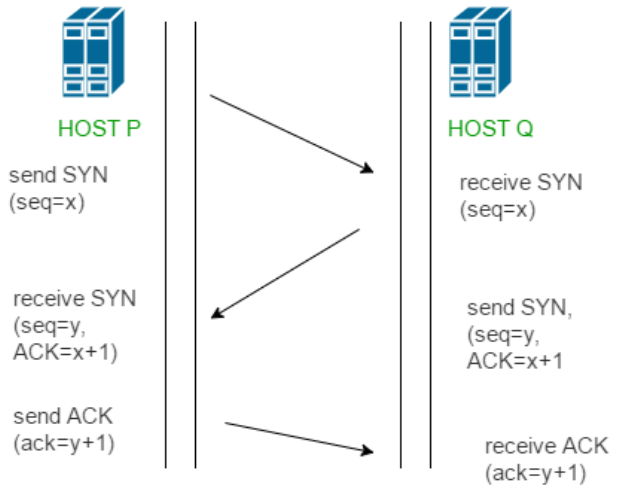

FIGURE 2: TCP THREE-WAY HANDSHAKE [1].

The client and the server received an acknowledgment when these steps are done, both can send data to each other. With the big amount of the connection is established every time it put the server under a lot of pressure, for example when a request for a small image or file, the client and the server must go through the whole process again to fulfill the request, again it's a drag for the network.

1.3 Moving to Information-Centric Networking

As we saw previously, the classic internet doesn't fit today's needs with the increasing devices especially IoT and most of the applications are content-oriented, the data is what is needed regardless of where it comes from. A new approach to evolve the Internet architecture is needed that is no longer host-centric [2]. ICN it's the future internet research for building an infrastructure sufficient for today's use. "The increasing demand for highly scalable and efficient distribution of content has motivated the development of future Internet architectures based on named data objects (NDOs), for example, web pages, videos, documents, or other pieces of information. The approach of these architectures is commonly called ICN. In contrast, current networks are host-centric where communication is based on named hosts, for example, web servers, PCs, laptops, mobile handsets, and other devices" [3]. The main component of ICN are:

- Named Data objects
- Naming and Security
- Application Programming Interface
- Routing and Forwarding
- Caching

The other more recent projects that are being actively developed are Data-Oriented Network Architecture (DONA), Publish-Subscribe Internet Routing Paradigm (PSIRP), Network of Information (NetInf), and Content-Centric Networking (CCN), we will talk in more detail about CCN.

1.3.1 Content-Centric Networking (CCN)

Content-Centric Networking is considered an ICM, communication on CCN is based on named data instead of IP addresses. The characteristic of CCN is the exchange of content requests by messages called ‘interests’ and content return message called ‘content objects’. The goal of the CCN is to create an efficient architecture that matches today's communication problems, application design pattern, and at least scalable and efficient as TCP/IP and much more secure with the requirement of fewer configurations [4].

1.3.2 Content Model

In CCN, names are an identifier of information collections instead of information containers as in Figure 3. The name of information can have many different names as in web-like links.

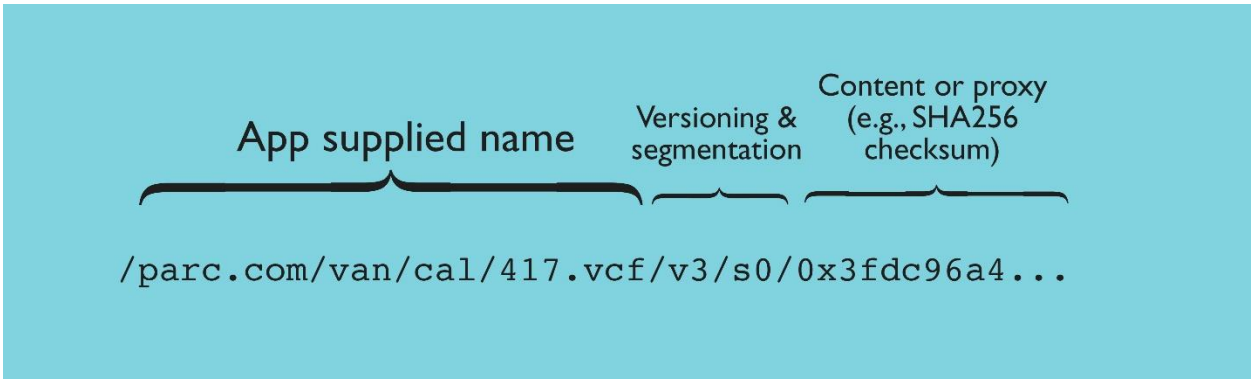

FIGURE 3: NAME IN CCN. [4]

1.3.3 CNN Packets

CNN uses two types of packets to communicates, one called ‘Interest’ packet similar to HTTP “get” and the other called data packet similar to the HTTP response, both are encoded in binary XML [4]. Figure 4 show the interest and the data packet and their attribute.

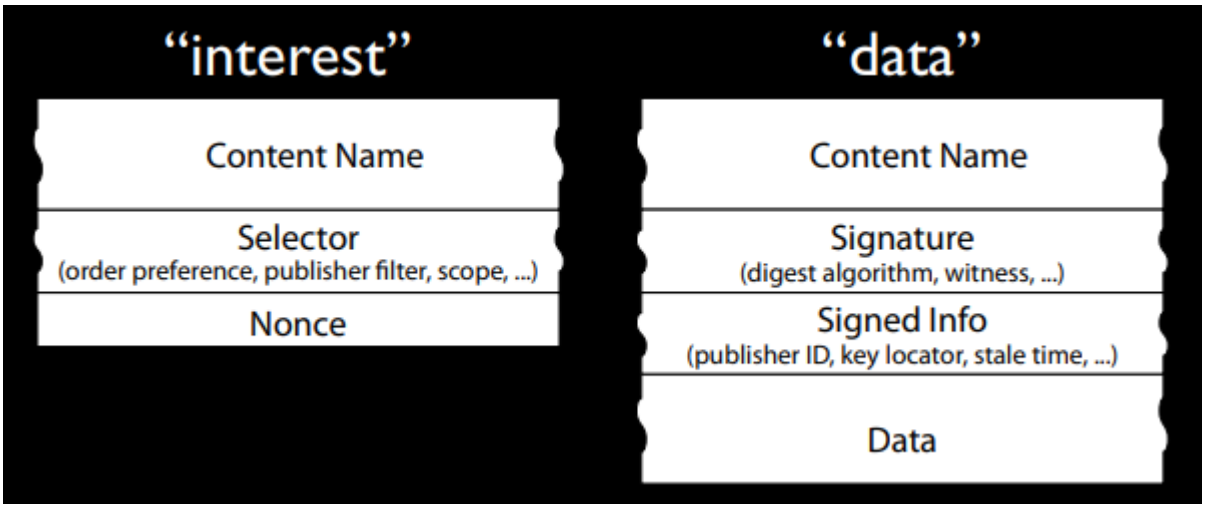

FIGURE 4: INTEREST AND DATA PACKETS [4].

"Longest Match" lookups are done using the hierarchal structure that helps guarantee log(n) state scaling for globally access data, the explicit structure of the CNN names allows efficient lookups as IP's [4].

1.3.4 Named Data Network

The other project that is currently researched from CCN is Named Data Networking. Below, Figure 5 describes the difference between NDN and IP structures.

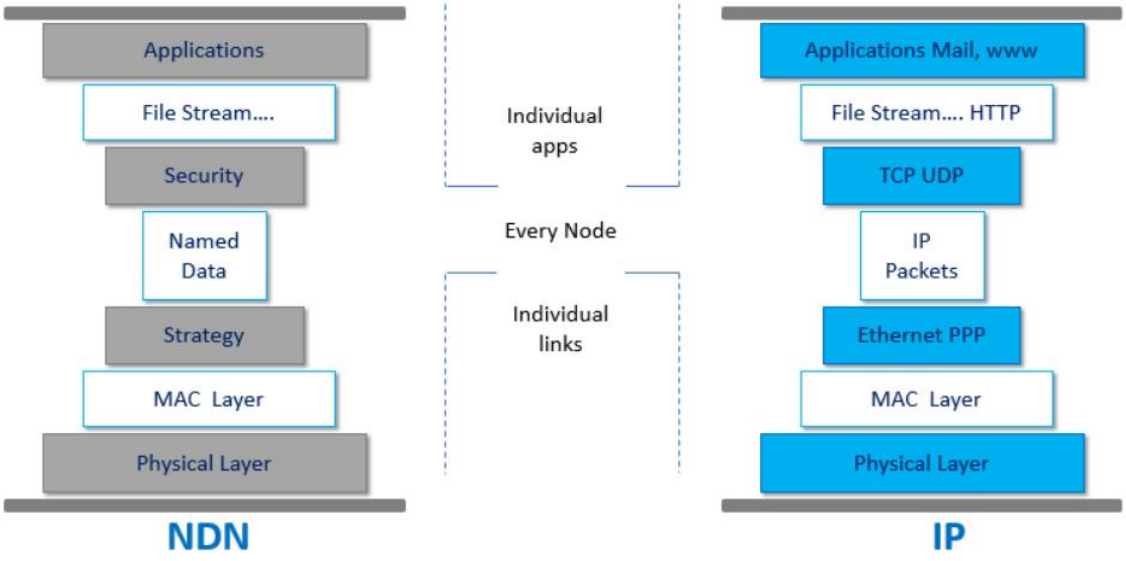

FIGURE 5: COMPARISON OF NDN AND IP ARCHITECTURE [5].

1.4 Named Data Network

Named Data Network is a proposed future Internet architecture and new improvement of Information-Centric Network (ICN), Figure 6 shows the NDN network in more detail and how interest is requested and how it satisfied. The difference between IP and NDN is that IP delivers packets to hosts based on numerical IP addresses and NDN fetches data by using names [6].

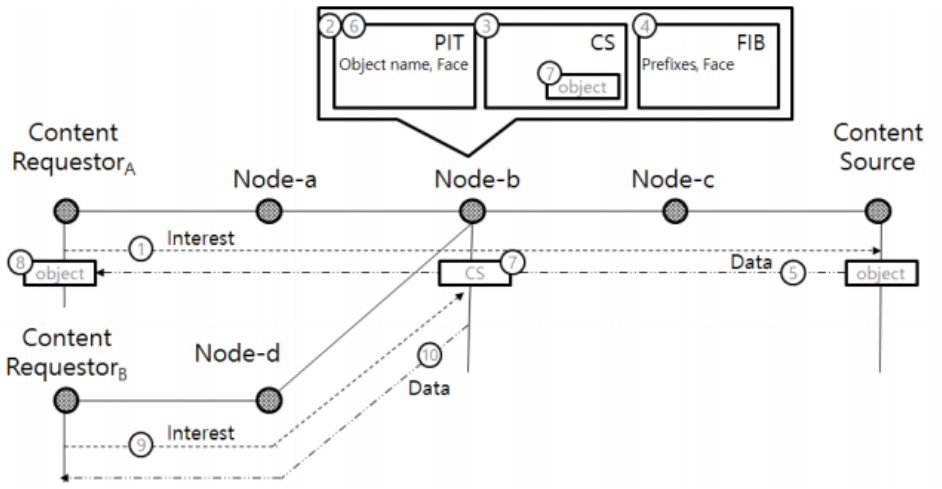

FIGURE 6: DATA NETWORKING ARCHITECTURE (NDN) INTEREST/DATA PROCEDURE [7].

1.4.1 NDN Structure

NDN nodes communicate through two types of packets: Interest packets and Data packets and every router in NDN have three types of data structure: Pending Interest Table (PIT), Content Store (CS), and Forwarding Information Base (FIB) The requester sent Interest packet to the network and if any router has the data in its cache it will send the Data to the requester. The Interest packet contains the name of the desire data and another parameter, the data packet contains the name and the data along with the signature of the producer of the data. When interest is received, the NDN router will look in the CS if a match is found then it will be sent to the requester. Otherwise, Interest will be looked up in the PIT if a PIT entry was found, the packet's incoming interface will be added to the list of the interface(s) and when the interest is available all the interested users will get the data packets. If no PIT entry is found then the Interest packet will be passed to the router's FIB which performs a longest prefix match (LPM) and when FIB entry is found for those LPMs, the Interest packet is forwarded to the next-hops and a new PIT entry is created with its incoming interface [6].

1.4.1.1 Content Store (CS)

In NDN the content store contains data packets, each data packet is identified by its Content Name (CN) and a producer signature, which allow multiple requesters (or consumers) to use the same stored data packet until it gets replaced by another data packet depending on caching policy, and because the Content Store has limited space capacity. Searching in the CS is done by exact matching of names, character by character [6].

1.4.1.2 Pending Interest Table (PIT)

When an interest packet arrives, a PIT entry is added. Each PIT entry has its fields Figure 7 CN, nonce, incoming and outgoing interface(s), and timer. The nonce combined with the CN gives us a unique identifier of an interest packet with the ability to detect loops and prevent duplications of packet forwarding. The outgoing interface(s) with the send-time help estimate the corresponding name prefix's Round Trip Time (RTT) to the performance of the interface. With every PIT entry added a timer is associated with it, the PIT entry will be deleted if no data packet returned till timer expiry. There are three main operations: insert a new PIT entry, delete a serviced PIT entry, and update an existing PIT entry. Updating an existing PIT entry means adding additional new interface(s) due to other requests to the same data that is already added to the PIT [6].

Name	**List of Nonce**	**List of incoming interfaces**		**List of Outgoing Interfaces**	
CN	Nonce	Interface ID	Timer	Interface ID	Send-time

FIGURE 7: STRUCTURE OF A PIT ENTRY [6].

1.4.1.3 Forwarding Information Base

FIB maintains next-hop(s) and other information for each reachable destination name prefix. The FIB is populated by the routing protocol and used for forwarding the interest packet upstream [6]. When an interest packet arrives at an NDN router it will be searched in the CS by its CN to find a match, if there was a hit, the corresponding data packet is returned to the requester if not, the CN is looked up in PIT, if PIT entry is found the operation of PIT entry update perform by adding the incoming interface(s) also known as *Interest aggregation*. If not the interest packet is passed to the router's FIB which perform a LPM, e.g., for a content name = *'/lagh/univ/course/pdf/f1'*, the possible LPMs found by FIB is *'/lagh'*, *'/lagh/univ'*, *'/lagh/univ/course'*, *'/lagh/univ/course/pdf'*, *'/lagh/univ/course/pdf/f1'*. The interest packet is forwarded to the newt hop(s) when a matching FIB entry is found and a new PIT entry is added with its incoming interface, otherwise, the interest packet is deleted or flooded to all outgoing interfaces depending on the forwarding policy. When a data packet arrives the PIT entries are searched for a matching CN. If there is a match, the data packet will be forwarded to all the listing of the incoming interface(s) corresponding to the PIT entry, after that the PIT entry is deleted, and the data packet is stored in the CS based on the local caching policy of the NDN router, if no PIT entries match is found, the data packet is discarded and dropped [6]. Figure 8 shows the process of forwarding in the NDN network.

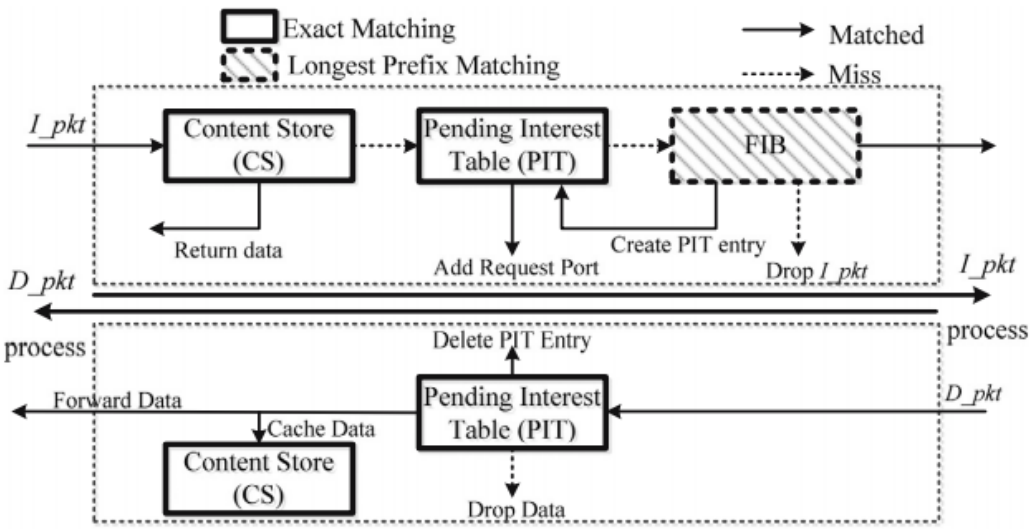

FIGURE 8: FORWARDING PROCESS AT NDN NODE [6]

1.4.2 Forwarding in NDN

In every router, there is three data structure as we mentioned before PIT, CS, and FIB. There is also a similar FIB in IP, the difference relies on the NDN that it contains name prefix instead of IP address. Also, there is a strategy module for incoming interest packet in an NDN router, determining what interface is used for forwarding interest is done by the strategy module, that makes forwarding decision adaptive to the network conditions. [5]. Forwarding in the NDN plane can detect failures and performs recoveries.

1.4.2.1 Scalable Forwarding

The NDN forwarding plane is supported by the three structures of PIT, CS, and FIB, also the forwarding plane that is used in the current CCNx implementation. In the intelligent forwarding strategies, the packets can have multiple outgoing interfaces basing on FIB lookup results and considering the network environment to perform a forwarding decision [8].

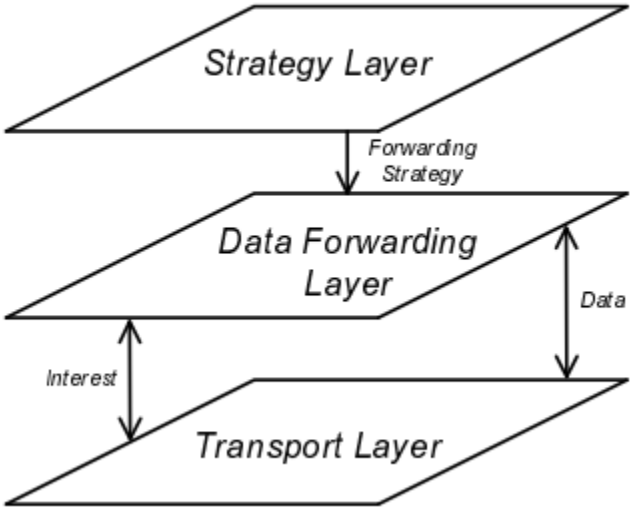

FIGURE 9: LAYERS OF THE FORWARDING PLANE [8].

NDN name lookups can happen in PIT, CS, or FIB by both longest prefix match and exact string match. Content distribution performance is influenced by the improved cache-hit rates, and that depends on an effective caching replacement policy.

1.4.3 Security in NDN

In any networking architecture, security is the most requirement that must be met. “The key challenges of NDN security are cost-effective security operations, trust management, and privacy protection” [6].

In every data packet, a trusted content must have the requirement of data integrity, origin authentication, and relevance of the data. The publisher of the data sign the content with one of the signature algorithm (like RSA, DSA, or EC-DSA), each signed data packet contains [9]:

- Signature: a public key that is needed for the verification (like RSA, DSA)
- KeyLocator: references the key needed to verify the content signature. The field contains one of the following: public key, a certificate containing verification key, or NDN name referencing verification key.
- PublisherPublicKeyDigest: it’s the hash of the content producer’s public key.

The trusted server or directories of the content doesn’t need access control policies, because NDN uses content encryption and confidentiality to support access control, and the decryption key can be passed along with the content, so there is no need for distributed decryption key [6].

1.4.3.1 Physical & data link layer

For securing the Physical and the data layer against the two major attacks, networking sniffing and man-in-middle attack, NDN design an encryption mechanism similar to TCP/IP architecture.

1.4.3.2 DNS Attack

In a DNS attack, the attacker re-routes the traffic to the targeted server, and that is not possible in NDN because re-routing the path between the source and the receiver doesn't make an effect if the received content is valid [6].

1.4.3.3 DoS and DDoS attacks

Those are the major two attacks, denial of service and distributed denial of service are commons attacks that became very effective and they are easy to initiate. In DDoS, the attacker control bots or machines also called zombies, and they flood the victim network by sending streams of packets (TCP, UDP, or ICMP) and saturates the network or the victim resources and the point is to overwhelm the victim with traffics.

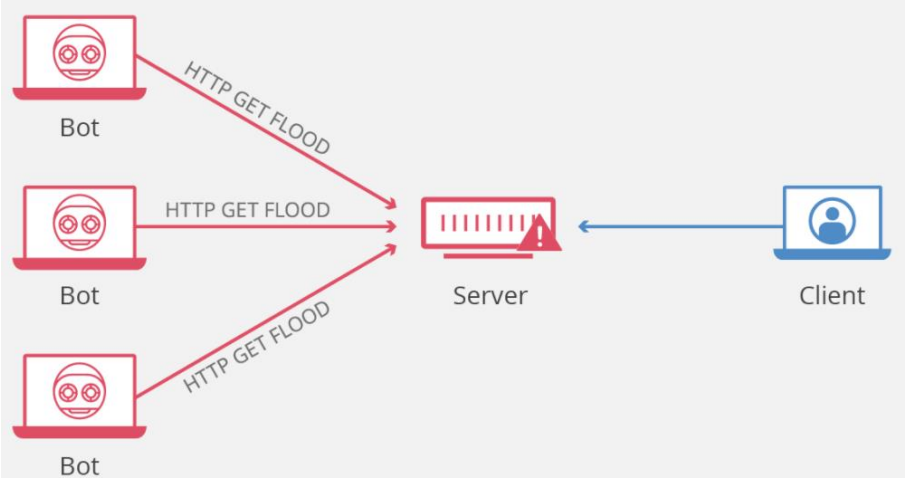

FIGURE 10: DDOS ATTACK. IMAGE SOURCE: CLOUDFLARE.COM

In NDN this attack is limited, once the requested data is pulled, it will be cached in an NDN router, so the requested interest will be limited for reaching the victim.

1.4.3.4 Privacy

In NDN, content encryption and its name are both needed for providing privacy. In IP both the content and the consumer are known, but in the NDN the content is known but the consumer is unknown, it's hard to know how requested the data without a direct connection, the only weak point is in the NDN router because it contains the user's requests, therefore the NDN require privacy at the router level [6].

1.4.4 Caching

The cache is a data storage with a high-speed that stores data, the main goal of the cache is to improve data retrieval performance. The data is stored in speed storage such as RAM (Random Access Memory), the limit of the size pushes to the decision of what data is worth to store or discard, and for that, we use caching management strategy.

NDN uses in-network caching, that is every intermediate node in the network cache content in its CS and act as a content provider that increases the latency and the speed of the network for the consumer, and lift the pressure on the content publisher or server, the caching decision is made by policies, and there is two types of caching management policy, placement and replacement that we will discuss in more details.

1.5 Cache Management

Caching also known as in-network storage is an essential component in NDN, every NDN router or device in the network has a CS, the caching management strategy that it's used affects the performance of the network. Caching can improve both the consumer by decreasing access time latency and reduce the loads on the producer. "There are two aspects of the caching process that caching strategies can affect: The caching decision and the cache replacement decision" [10].

1.5.1 Placement Policy

Placement policy is a caching decision, the router chooses to store data when it arrives, depending on many factors or metrics that data is stored in the CS or discarded. There is different Placement policy like:

- **Cache Everything Everywhere (CEE)** that cache every incoming Data packet that is not in its content store may also translate to the low diversity of data and decrease the hit ratio.
- **Leave Copy down (LCD)** content is always cached only at the next hop from the node where the cache hit occurred (i.e. the first node to receive the Data) and nowhere else [10]. Figure 11 shows an example.

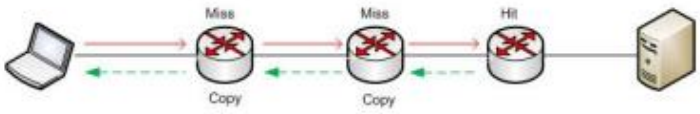

FIGURE 11: LEAVE COPY EVERYWHERE. [11]

1.5.1.1 Coordination-based schemes

In coordination-based schemes, caching nodes/routers in the network exchange information and apply caching decision with other each other. That reduces content redundancy and improves the cache diversity, it also affords control overhead over the network. Based on that we can divide these schemes into:

1.5.1.1.1 Explicit Cache Coordination Schemes

In an explicit cache coordination scheme, the nodes are prerequisite with the information of the used network topology, cache's state, and the user's access frequency [12]

1.5.1.1.2 Implicit Cache Coordination Schemes

In an implicit cache coordination scheme the caching node doesn't have prerequisite information, instead, they exchange small information among the caching router to apply the caching decision of certain content to where to be cached [12].

1.5.2 Replacement Policy

It is the decision the router chose when a data packet arrives and the content store is full, the decision is made for choosing what data in the content store to be replaced. We have many strategies for replacement like Least Recently Used (LRU), Least Frequently Used (LFU), and First In First Out (FIFO). Some strategies take to concern the mobility of the nodes and put into consideration other factors like mentioned in the [13].

1.5.2.1 Least Recently Used (LRU)

The least recently used replacement policy is simple to implement, it is based on the recent and the timestamp of the content, when the CS reaches the maximum capacity it replaces the content that has not been used for a long time with the new arrived content. LRU is used when the CS is full and content needs to be replaced, otherwise, the content is inserted into the content store.

LRU algorithm:

- If the cache is not full then the item is saved cached
- Else, the cache is full: find the least recently used item and replace it with the arrived new item.

Example of LRU:

Let say we have 7 requests and the CS with a size of 3

Requests	7	0	3	1	0	1	5
	7	7	7	1	1	1	1
		0	0	0	0	0	0
			3	3	3	3	5

FIGURE 12: LRU EXAMPLE

As Figure 12, the sequence of requested data is [7, 0, 3, 1, 0, 1, 5], the size of the cache is 3. When the first 3 elements arrive at the node, all of them get saved in the cache. When 1 comes, the 7 is replaced because it the least used element, and so on.

1.5.2.2 Least Frequently Used (LFU)

In the LFU the content with the lowest frequency is replaced, this policy is also simple to implement, by adding a counter to the content that is stored and increase it every time that content is requested.

LFU algorithm:

- Each item stored in the cache has a counter (initial value is 0) that increased depending on the times the item is accessed.
- If the cache is not full then the newly arrived item is stored.
- Else, the item with the minimal counter is replaced with the newly arrived item.

Example of LFU:

We use the same data we used in the previous example of LRU:

The data and its frequency (data, frequency)

Requests	7	0	3	1	0	1	5
	(7,1)	(7,1)	(7,1)	(1,1)	(1,1)	(1,2)	(1,2)
		(0,1)	(0,1)	(0,1)	(0,2)	(0,2)	(0,2)
			(3,1)	(3,1)	(3,1)	(3,1)	(5,1)

FIGURE 13: LFU EXAMPLE

In Figure 13, we use the same sequence as before, but with the initial value of what it called Frequency of 0. Every time an existent element is requested; that is a hit and the frequency value of that element is incremented by 1. When the frequency is the same for all the stored content we apply FIFO.

1.5.2.3 First In First Out (FIFO)

FIFO is very straightforward we replace the oldest content that was put into the CS with the new one that arrives.

Example if FIFO:

Requests	7	0	3	1	0	1	5
	7	7	7	1	1	1	1
		0	0	0	0	0	0
			3	3	3	3	5

FIGURE 14: FIFO EXAMPLE

In Figure 14, the oldest element that is inserted will be replaced by the new element that is arrived when the cache is full.

1.5.2.4 Random replacement strategy

When a data packet arrives the router that uses a Random Replacement Strategy will choose the content from the CS randomly. Memory and replacement time is less due to the absence of criteria of selection.

1.5.2.5 Cache Content Replacement (CCR)

[14] Propose a Cache Content Replacement that takes into consideration the estimated content local and global popularity for placement, and take the instantaneous hit rate of content with other criteria to choose what content to replace.

1.6 Conclusion

As we saw in this chapter the classic internet has its pro and cons, but in the modern days, we must shift to a new architecture that focuses on the content rather than the source and the distention of it. We talk about the internet protocol and its features, we talked about the new architecture ICN and its structure and models. In the ICN a new improvement come to it and we got the NDN, that focus on the data and identifying it by its name. in the next chapter we will

talk about machine learning and its types, we also we will see the apriori algorithm that we will use in our method.

2 Machine Learning

2.1 Introduction

In this chapter, we are going to talk about Machine Learning and its types, and see some examples of a popular algorithm that is used. We will see also the Apriori algorithm and explain in detail how it works, then we will finish with an example.

2.2 Machine learning

Machine learning (ML) is a branch of Artificial Intelligence (AI) that automatically learns and improves from experience, the idea is that the machine can learn from the data that it's given to it, it can identify the pattern and make decisions. With the momentum that Machine Learning and AI get from giant corporations and the massive amount of investment that have made a big push on the field, as we see in all products and technologies, it's essential to take advantage of that for gain in profit and performance.

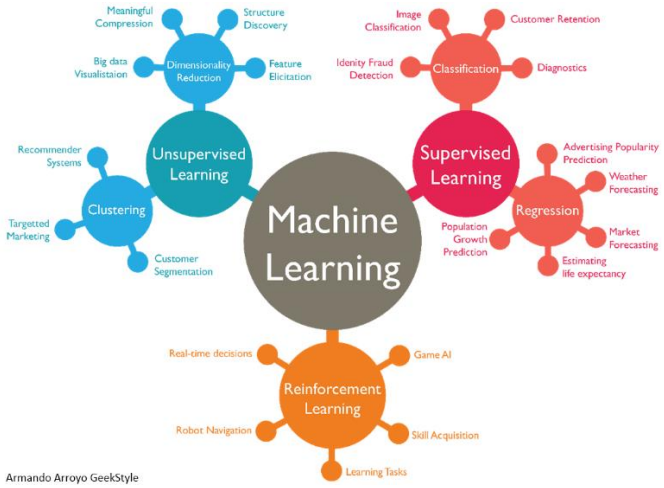

FIGURE 15: MACHINE LEANING BRANCHES [15].

There are types of ML, the widely known four types are Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning, and we will talk about them in more detail next.

2.2.1 Unsupervised Learning

Is used against data that has no historical labels. The system is not told the "right answer." The algorithm must figure out what is being shown. The goal is to explore the data and find some structure within it. Unsupervised learning works well on transactional data. For example, it can identify segments of customers with similar attributes who can then be treated similarly in marketing campaigns. Or it can find the main attributes that separate customer segments from each other. Popular techniques include self-organizing maps, nearest-neighbor mapping, k-means clustering, and singular value decomposition. These algorithms are also used to segment text topics, recommend items and identify data outliers [16]. "Unsupervised learning is commonly used for finding meaningful patterns and groupings inherent in data, extracting generative features, and exploratory purposes." [17].

One of the most popular clustering algorithms that fall in the category of unsupervised learning:

- K-means Clustering Algorithm
- K-NN (k nearest neighbors)
- Singular Value Decomposition

2.2.2 Semi-supervised Learning

Is used for the same applications as supervised learning. But it uses both labeled and unlabeled data for training – typically a small amount of labeled data with a large amount of unlabeled data (because unlabeled data is less expensive and takes less effort to acquire). This type of learning can be used with methods such as classification, regression, and prediction. Semi-supervised learning is useful when the cost associated with labeling is too high to allow for a fully labeled training process. Early examples of this include identifying a person's face on a webcam [16].

2.2.3 Reinforcement Learning

Robotics, gaming, and navigation are all examples of reinforcement learning. It uses used to figure out which activities result in the most rewards through trial and error. The agent (the learner or decision-maker), the environment (everything the agent interacts with), and actions are the three main components of this sort of learning (what the agent can do). The goal is for the agent to make a decision. [16].

An example of a reinforcement learning algorithm is Q-learning, the “Q” stands for Quality, meaning that how useful a given action is in gaining some future reward.

Q-learning

The goal of Q-learning and related algorithms is for an agent to learn the best policy from its previous interactions with the environment, the history of an agent is a sequence of state-action-reward [18].

$$< s0, a0, r1, s1, a1, r2, s2, a2, r3, s3, a3, r4, s4 \ldots >,$$

Here meaning that the agent was in the state s_0 and did action a_0, which gave a result of receiving the reward r_1 and being in the state s_1, after that it did action a_2, received reward r_3 and ended up in the state s_3 and so on [18]. This history of interactions is treated as a sequence of experiences, and the experience is a tuple: $< \mathrm{s}, \mathrm{a}, \mathrm{r}, \mathrm{s}' >$,

This means the agent was in state s, did action a, received reward r and went to state s' [18].

Q-learning algorithm

```
controller Q-learning(S, A, γ, α)
    Inputs
        S is a set of states
        A is a set of actions
        γ the discount
        α is the step size
    Local
        real array Q[S, A]
        previous state s
        previous action a
    initialize Q[S, A] arbitrarily
    observe current state s
    repeat
        select and carry out an action a
        observe reward r and state s'
        Q[s, a] ← Q[s, a] + α (r + γ max_a' Q[s', a'] − Q[s, a])
        s ← s'
    until termination
```

Example:

- States: $s_0, s_1, s_2, s_3, s_3, s_4, s_5$
- Actions:
  - right
  - left
  - up (0.8 up, 0.1 left, 0.1 right)
  - upC (“up carefully”, always up, but costs reward -1)
  - additional rewards:
    - Out of bounds -1

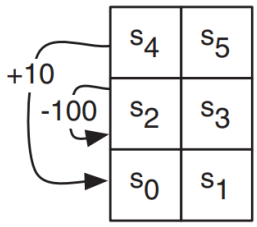

FIGURE 16: THE ENVIRONMENT OF A TINY REINFORCEMENT LEARNING

s	a	r	s'	Update
s_0	upC	-1	s_2	$Q[s_0, upC] = -0.2$
s_2	up	0	s_4	$Q[s_2, up] = 0$
s_4	$left$	10	s_0	$Q[s_4, left] = 2.0$
s_0	upC	-1	s_2	$Q[s_0, upC] = -0.36$
s_2	up	0	s_4	$Q[s_2, up] = 0.36$
s_4	$left$	10	s_0	$Q[s_4, left] = 3.6$
s_0	up	0	s_2	$Q[s_0, upC] = 0.06$
s_2	up	-100	s_2	$Q[s_2, up] = -19.65$
s_2	up	0	s_4	$Q[s_2, up] = -15.07$
s_4	$left$	10	s_0	$Q[s_4, left] = 4.89$

FIGURE 17: SEQUENCE $< s, a, r, s' >$ EXPERIENCES, AND THE UPDATE, WHERE $\gamma = 0.9$ AND $\alpha = 0.2$ [18].

2.2.4 Supervised Learning

The formal supervised learning process involves input variables, which we call (X), and an output variable, which we call (Y). We use an algorithm to learn the mapping function from the input to the output [17] The learning algorithm receives a set of inputs along with the corresponding correct outputs, and the algorithm learns by comparing its actual output with correct outputs to find errors. It then modifies the model accordingly. Through

methods like classification, regression, prediction, and gradient boosting, supervised learning uses patterns to predict the values of the label on additional unlabeled data. Supervised learning is commonly used in applications where historical data predicts likely future events. For example, it can anticipate when credit card transactions are likely to be fraudulent or which insurance customer is likely to file a claim [16].

The machine learning model learns to adjust the mapping of the example input entities with their associated labels, When the models are trained with these examples, we use them to make new predictions on unknown data [17].

An example of a supervised learning algorithm is:

- Support Vector Machine (SVM)
- Nearest Neighbor
- Decision Trees
- Linear Regression
- Neural Networks

2.2.5 Machine learning in Cache

The approach of caching like LRU, LFU, FIFO, and RR, although simple, can't handle the dynamics of content popularity and network topologies. Recent efforts and needs have turned toward Machine Learning (ML), Deep Learning (DL), and optimizations approach [19].

2.3 Apriori Algorithm

Apriori algorithm is an algorithm for frequent itemset mining and for finding association rule, it was proposed by Rakesh Agrawal and Ramakrishnan Srikant in 1994. The Apriori algorithm can help us find the association rule in data and item sets. The name *apriori* is based on the fact that it uses the prior knowledge of frequent itemsets. Apriori employs an iterative approach known as a level-wise search, where *k*-itemsets are used to explore (k + 1)-itemsets. First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item and collecting those items that satisfy minimum support. The resulting set is denoted by L_1. Next, L_1 is used to find L_2, the set of frequent 2-itemsets, which is used to find L_3, and so on, until no more frequent k-itemsets can be found. The finding of each L_k requires one full scan of the database [20].

Let Σ be an alphabet of *n* item $\Sigma = \{a_1, a_2, a_3 \dots a_n\}$ [21].

A transaction is an itemset of Σ

$T \subseteq \Sigma\ T_i \rightarrow I = \{a_1, a_2, a_3\}$

An itemset I is a subset of T_i

$I \subseteq T_i \rightarrow I = \{a_1, a_2\}$

A transaction Dataset D is a set of n transactions and N is the number of transactions in D

$D = \{T_1, T_2, T_3, T_4, \dots, T_n\} \rightarrow N = |D|$

2.3.1 Support

Support is the popularity of an item in transactions, we calculate the support of item *A* by dividing its appearance by the number of transactions.

An example of 10 transactions with item *A* that appear 3 times in transactions, with the minimum support count of 2, we put 2 as minimum support so min_sup = 2 so the corresponding relative support is 2/10 = 20%.

Support(*A)* = *appearance of A in all transactions / number of transactions*

Support(*A)* = 3 / 10 = 30%

2.3.2 Confidence

Confidence is that a user requesting item *A* will possibly request item *B*. we calculate that by dividing the number of transactions that contain both items *A* and *B* by the number of transactions containing *A*.

Example of confidence: let's say we have 2 transactions containing both *A* and *B*, and the number of the transaction containing A is 3.

A → B = Confidence = 2 / 3 = 66%

2.3.3 Lift

The lift is the increase ratio in the request of *B* when *A* Is requested. The formula for calculating the list is:

Lift = Confidence(A → B) / Support(A)

We will take the previous example results to calculate the lift of *A* and *B*.

Lift = 66 / 30 = 2

We say that the item *A* and *B* are twice likely to be requested together than the item *A* is requested alone.

2.3.4 Example of Apriori algorithm

Every transaction has its id let say it called TID, in this table-1 we have the data of users requests of items:

TID	Items
T1	1 3 4
T2	2 3 5
T3	1 2 3 5
T4	2 5
T5	1 3 5

1) In the first iteration we will add the support value of 2 and calculate the support value of each item in the sets:

TID	Items
T1	1 3 4
T2	2 3 5
T3	1 2 3 5
T4	2 5
T5	1 3 5

C1

Itemset	Support
{1}	3
{2}	3
{3}	4
{4}	1
{5}	4

Item 4 has support less than the min support 2, so we will remove it in the next iterations. We have now the final table F1.

C1

Itemset	Support
{1}	3
{2}	3
{3}	4
{4}	1
{5}	4

F1

Itemset	Support
{1}	3
{2}	3
{3}	4
{5}	4

2) In the second iteration we will create itemsets of size 2 and calculate their support values. The items in F1 are used in this iteration.

TID	Items
T1	1 3 4
T2	2 3 5
T3	1 2 3 5
T4	2 5
T5	1 3 5

C2

Itemset	Support
{1,2}	1
{1,3}	3
{1,5}	2
{2,3}	2
{2,5}	3
{3,5}	3

F2

Itemset	Support
{1,3}	3
{1,5}	2
{2,3}	2
{2,5}	3
{3,5}	3

We eliminate itemsets that have support less than the minimal, {1,2} has the support of 1 so it's removed.

Pruning: using the itemsets of C3 and dividing them into subsets with the elimination of the subsets that have a support value less than the minimum support threshold of 2.

TID	Items
T1	1 3 4
T2	2 3 5
T3	1 2 3 5
T4	2 5
T5	1 3 5

C3

Itemset	In F2
{1,2,3} ,{1,2} ,{1,3} ,{2,3}	NO
{1,2,5} ,{1,2} ,{1,5} ,{2,5}	NO
{1,2,3} ,{1,5} ,{1,3} ,{3,5}	YES
{2,3,5} ,{2,3} ,{2,5} ,{3,5}	YES

3) In the third iteration we will eliminate {1,2,3} and {1,2,5} because both contain {1,2}.

TID	Items
T1	1 3 4
T2	2 3 5
T3	1 2 3 5
T4	2 5
T5	1 3 5

F3

Itemset	In F2
{1,3,5}	2
{2,3,5}	2

4) In the fourth iteration we will use F3 to create C4:

TID	Items
T1	1 3 4
T2	2 3 5
T3	1 2 3 5
T4	2 5
T5	1 3 5

F3

Itemset	In F2
{1,3,5}	2
{2,3,5}	2

C3

Itemset	In F2
{1,2,3,5}	1

The final itemset of {1,2,3,5} have the support of 1, and it's less than the min support, we will stop here and take F3 as the final itemsets:

For I = {1,3,5} the subsets are {1,3},{1,5}, {3,5}, {1}, {3}, {5}

For I = {2,3,5} the subsets are {2,3},{2,5}, {3,5}, {2}, {3}, {5}

2.3.4.1 Applying rules

We will apply rules on the itemsets of F3. In this example we set the Confidence minimum to 60%.

For every subset S of I, we will output the rule:

- S -> (I-S) that means S recommends I-S
- If Support(I) / Support(S) >= min_conf value

{1,3,5}

Rule 1: {1,3} -> ({1,3,5,} – {1,3}) means 1 & 3 -> 5

Confidence = Support(1,3,5) / Support(1,3) = 2 / 3 = **66% > 60%**

The **rule 1** is selected.

Rule 2: {1,5} -> ({1,3,5} - {1,5}) means 1 & 5 -> 3

Confidence = Support(1,3,5) / Support(1,5) = 2/2 = **100% > 60%**

The rule 2 is selected.

Rule 3: {3,5} –> ({1,3,5} – {3,5}) means 3 & 5 –> 1

Confidence = Support(1,3,5) / Support(3,5) = 2/3 = **66.66% > 60%**

The **rule 3** is **selected.**

Rule 4: {1} –> ({1,3,5} – {1}) means 1 –> 3 & 5

Confidence = Support(1,3,5) / Support(1) = 2/3 = **66.66% > 60%**

The **rule 4** is selected.

Rule 5: {3} –> ({1,3,5} – {3}) means 3 –> 1 & 5

Confidence = Support(1,3,5) / Support(3) = 2/4 = **50% < 60%**

The **rule 5** is rejected.

Rule 6: {5} –> ({1,3,5} – {5}) means 5 –> 1 & 3

Confidence = Support(1,3,5) / Support(5) = 2/4 = **50% < 60%**

The **rule 6** is rejected.

2.4 Conclusion

In this chapter, we talk about Machine Learning and its potential for solving a large of simple and complex problems. Supervised, unsupervised, semi-supervised, and reinforcement learning are all machine learning types that can be suited for different types of fields and problems. We also talked in the end about the apriori algorithm and its capability of helping in increasing the performance of cache management in this work approach. In the next chapter we will represent our approach of using machine learning in cache management, we will talk about the dataset and the simulation that we did.

3 Apriori-Based Cache Management

3.1 Introduction

Todays need to obligate the use of cache management that is adaptable and fit to different scenarios, one of the options to solve this problem is by using a machine learning approach. We suggest the Apriori algorithm; supervised learning, integrated into the cache policy for increasing cache hits and also decreasing the waiting time for data. In this chapter, we talk about the objective, method of the work, the tools we used, the steps of the processes for achieving the goal of implementing the suggested approaches, and finally the result with its interpretations.

3.2 Apriori-Based Recommendation

In this section, we will talk about the datasets, treatments, and tools for that treatment.

3.2.1 Datasets and treatment

After we obtained the data of real traffic from a website [22] that we have access to, as a log for requested URLs in a period of time Figure 18, we treat the file to be suited for input to the used software.

114.119.140.29	/tag/52/	200	HTTP/1.1
18.118.144.241	/	200	HTTP/1.1
66.249.66.53	/ads.txt	200	HTTP/1.1
66.249.66.49	/robots.txt	200	HTTP/1.1
114.119.146.38	/tag/44/	200	HTTP/1.1
114.119.146.38	/tag/44	301	HTTP/1.1
114.119.141.178	/tag/53/	200	HTTP/1.1
54.236.1.13	/	200	HTTP/1.1
54.236.1.13	/robots.txt	200	HTTP/1.1
54.236.1.13	/explore/	200	HTTP/1.1
54.236.1.13	/robots.txt	200	HTTP/1.1
114.119.151.17	/tag/55/	200	HTTP/1.1

FIGURE 18: REQUESTED LINKS

3.2.1.1 Pre-treatment of data

To begin with, we filter the requested links in the obtained file, as it will represent the interest names that we will use for our simulation. The list of links is the packets that it will be sent by a client to an edge node in the NDN network. We treat the file to be compatible be a CSV file, and be accepted for the Weka software to treat.

3.2.1.2 Association rules - Weka

To find the association rules that help us find the probability of relationships between items of data. We use a software called Weka [23], the software is a collection of machine learning algorithms, and it contains the tools for association rules mining that we need in our experiment. Figure 19 shows the interface of the software Weka.

FIGURE 19: WEKA GUI INTERFACE [23].

The input we provide to Weka is the log file that we mentioned before, it can be fed as a file of type CSV or ARFF. Figure 20 shows the interface of setting the attributes of the algorithm.

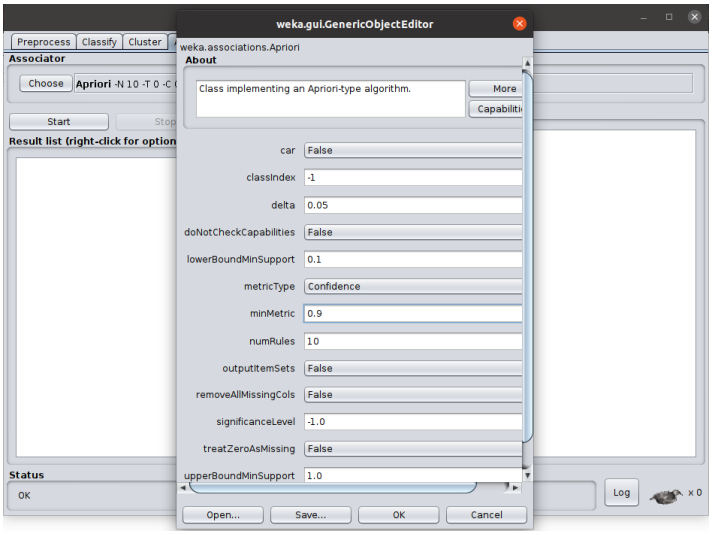

FIGURE 20: WEKA APRIORI-ALGORITHM [23].

In Figure 21 we see the output of the association.

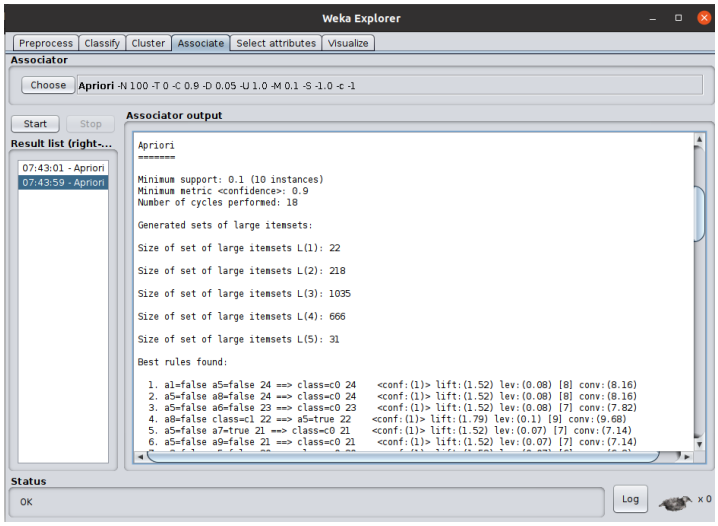

FIGURE 21: WEKA ASSOCIATION OUTPUT [23].

3.2.2 NS-3 simulator – ndnSIM

Due to the new architectures that are not implemented today, experimenting may not be possible or difficult to perform and control, here the simulators can offer great help and be the best choice to research and experiment with.

NS-3 [24] is a network simulator for Internet systems, targeted mainly for research and education use. It's free-to-use software. NS-3 includes models of how packet data networks work and perform, as well as a simulation engine that allows users to run simulations. To do research that is more difficult or impossible to perform with real systems, to investigate system behavior in a highly controlled, reproducible setting, and to understand how networks work [24].

The ndnSIM is an open-source simulation platform based on the NS-3 simulator framework. The major NDN primitive can be implemented using the ndn-cxx library that also can be used to implement various applications. The modular NDN Forwarding Daemon (NFD) is a network forwarder, its functionality to Interests and Data packets [25].

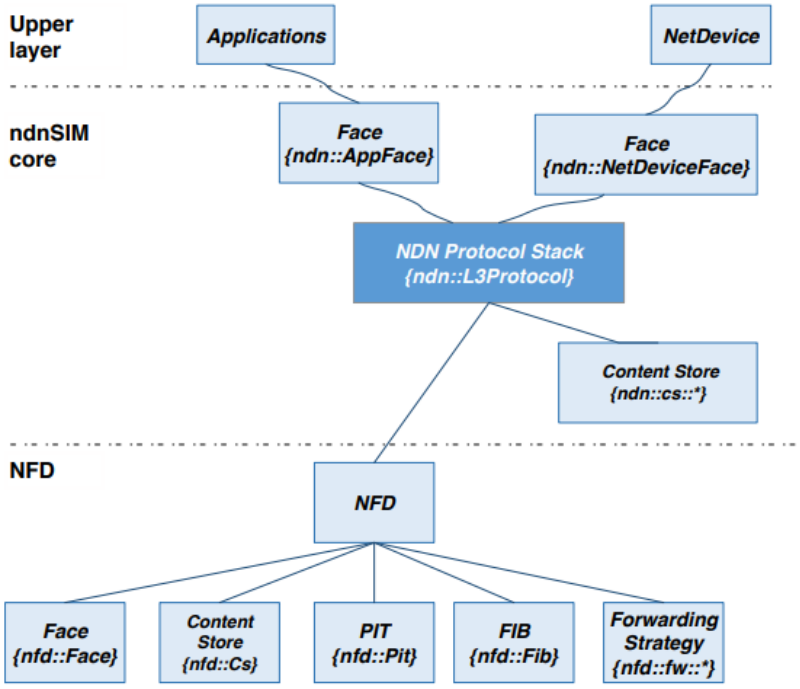

FIGURE 22: STRUCTURAL DIAGRAM OF THE NDNSIM DESIGN COMPONENT [25].

The ndnSIM structure is shown in Figure 22.and describes it more.

3.2.3 Objectives

In NDN, caching is an essential component. Every time the requester sends an interest, that interest can be found in some intermediate node if note it will be retrieved from the publisher it, our method aims to increase the hit ratio of the cache, by recommendation of the possible next requested interest(s) in the edge node of the NDN network decreasing client access latency, our method must sustain the number of the packets sent by a node to avoid congestion and network flooding. To achieve the aforementioned we use the Apriori algorithm.

3.2.4 Methodology

Providing the possibly next requested data in the edge node can offer an increased cache hit ratio of the network, low access latency, and reduce the load on the publisher. In this work, we considered the edge node is responsible for the recommendation of the next possibly requested data by using the association rules that it has. To avoid overwhelming the network and network congestion, we use the minimum confident parameter that is suited for our network, increasing the minimum confidence will produce a high chance of the recommended requested interest being meet with the client requested interest; in short, the minimum confidence is the interest requesting percentage, 0.9 minimum confident is 90% certainty that a data will be requested next.

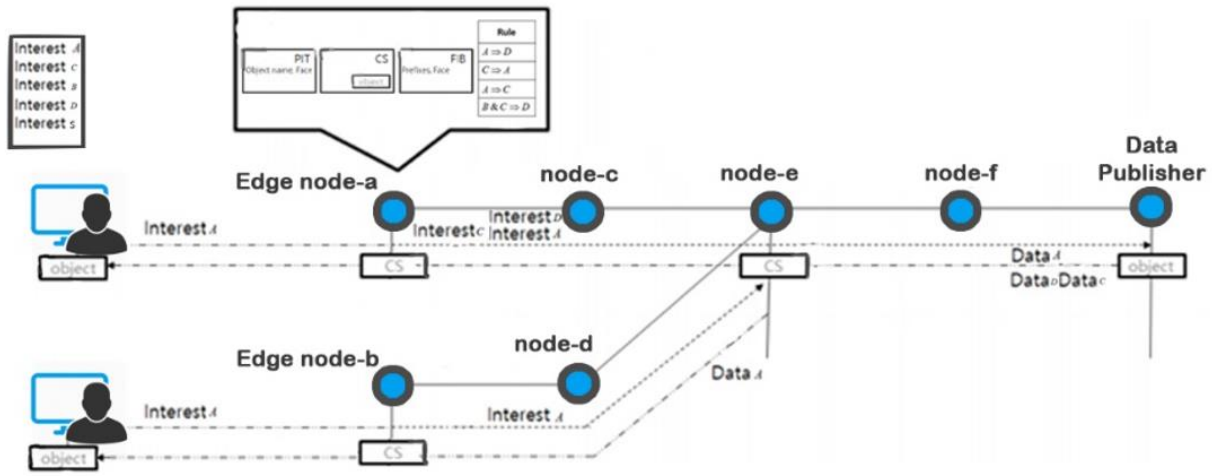

FIGURE 23: RECOMMENDING INTERESTS AT EDGE NODE LEVEL

As Figure 23 above shows, when a client requests an interest-A, the edge-node-a will do all the action that we mentioned before in NDN Structure, besides that it will do look up at the Association rules that it had, and see the rule for interest-A and find that interest-B and interest-C have a possible chance of being requested after interest-A, node-a will send interest for A and B if they are not stored in the cache (CS), when the data come back, it will be stored in the intermediate node also. The coming data being cached is depending on the caching policy that is implemented, we used in our work the LRU caching policy.

3.3 Experiment Environment

To simulate and validate our recommendation approach we use ndnSIM-2.7 [25]. This section will discuss the steps and the parameters for this simulation.

3.3.1.1 Simulation parameters settings

In this simulation every node in the network has the CS installed, and the Best Route Strategy is the default configuration of ndnSIM in all NDN nodes. In this work, we use LRU as a cache replacement strategy compared to the LRU-Recommendation approach. The parameters used are described in Table 1. These parameters are applied to all nodes in the NDN network.

TABLE 1: SIMULATION PARAMETER SETTING

Parameter	Value
Request Rate	50 req/s
CS size	20/40/80
Cache Strategy	Always
Forwarding Strategy	BestRoute
Simulation Time	1000 s
Number of topology nodes	8

As we saw in Table 1 the parameters, the request rate is 50 interests a second with the caching strategy always set. The forwarding strategy is BestRoute, the simulation was in different content store sizes of 20, 40, and 80 that runes for 1000s.

This work conducts simulation on the top-tree topology, it's described as a publisher connected with a router, this router is the way from the publisher to the two other routers, every one of those routers has two edge-node connected to it.

3.4 Performance Assessment

Here we will discuss the performance metrics and the results we achieve using this approach.

3.4.1.1 Evaluation Metrics

A cache hit is the metric for evaluating caches of NDN in this work. Every node have a variable to calculate cache hit and cache miss,

- Cache Hit: initially the value is 0. Every time a node received an interest packet and returns that data from its cache, the cache hit value is increased by 1.
- Cache Miss: also 0 initially, if a node received interest and it doesn't find the correspondent data in its local storage (the content store) then the cache miss value is increased by 1.

We use that to calculate the cache hit ratio of three different CS sizes 20-40-80.

$$\frac{\text{Number of cache hits}}{(\text{Number of cache hits} + \text{Number of cache misses})} = \text{Cache hit ratio}$$

3.4.1.2 Performance results

Figure 24, Figure 25, and Figure 26 show the comparison of the cache hit ratio between LRU and LRU-Recommendation with different content stores of 20, 40, and 80.

FIGURE 24: HIT RATIO - CONTENT STORE 20

Figure 24 illustrates the hit ratio percentage of LRU and LRU-Recommendation.

As we observe the hit ratio of both curves is not identical and they are below 30%, the LRU-recommendation is better than the LRU in the hit ratio. The LRU-Recommendation curve reaches more than 25% of hit ratio after ever 150s, with the same rat the LRU curve reaches almost 23% of hit ratio. At the time of the 750s, we see a spike in the hit ratio of the LRU-Recommendation by 27.7%. Because the CS size of 20 may cause the sudden periodic change of the cache hit ratio. The different performance of the two curves is explained by the cache hit ratio of the recommended interest being effective.

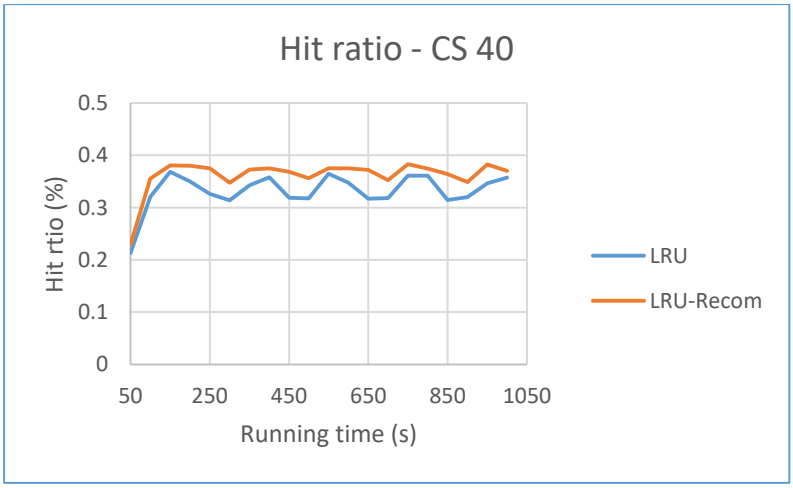

FIGURE 25: HIT RATIO - CONTENT STORE 40

Figure 25 represents the hit ratio percentage of the two curves of LRU and LRU-recommendation. We see a stable rate ranging closely between 37% and 35% of the cache hit ratio for LRU-R. The LRU curve shows less cache hit ratio and it drops to 31% and reaches 37% at some point in time. Increasing the content store affects the cache hit ratio for both methods.

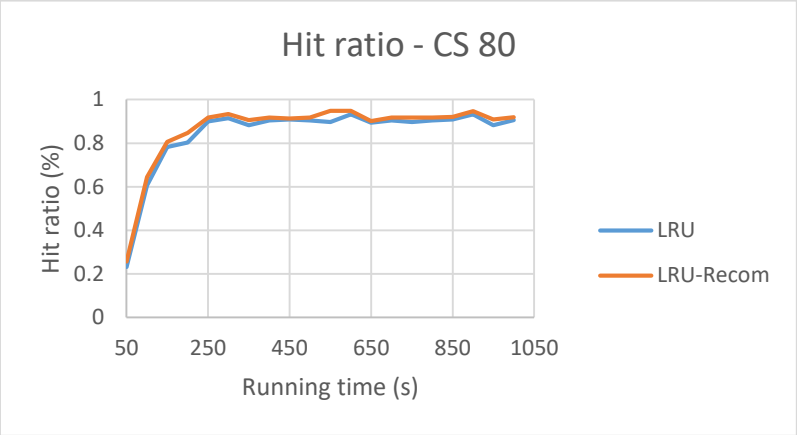

FIGURE 26: HIT RATIO - CONTENT STORE 80

In Figure 26 we observe that with a content store of size 80, we see the performance is high for both LRU and LRU-Recommendation and almost identical. The increase of the CS with the requested data that is provided is the logical explanation for these curves, there is a limit to the distinction of the interests that have been sent by the edge node in this experiment, causing the interest that is requested to meet the data in the cache with a large possibility of hits.

3.5 Discussion

As we saw before the results show better performance especially in the content store of sizes 20 and 40, meaning that the recommendation did increase the LRU by demanding the possibly next requested interest to be fetched and cached so the client will retrieve them from the cache rather than the content provider.

The result from the content store of size 80% is similar between the two, and that result is expected due to the limited distinction of the traffic that is used, forces the result, all the content is almost stored in the cache.

3.6 Conclusion

The new paradigm of the NDN network and its stateful data plane with caching allow us to introduce this approach of Recommendation using the Apriori algorithm. We present the data for this work to perform the simulation for this approach with real-world traffics. Our method increases the chance of the requested data by the client being stored in the cache in the intermediate node(s). The edge node is responsible for sending the recommendation interest. Our results demonstrate that we can increase the cache hit ratio. We did a treatment of data use it to output the association rules, after that, we implanted the rules at the level of the edge node in the NDN network to introduce the recommendation of interest.

Conclusion

In this work we discussed the network that runs today of IP and the source-destination architecture toward the Named Data Network, also we discussed the importance and the powerful things offered by Machine Learning to solve some of our problems. Afterward, we explained and gave examples of the Apriori algorithm and its associated rules that helped in our method. We presented an approach that increases the cache hit ratio, in turn, increases the overall performance of the network, and in the final chapter we described the approach and the methods.

The dataset of real traffics was treated to be fit to the tools we used. In addition, that data was fed as an input for the software we used to produce the considered association rules. Using these rules, the edge node can determent the recommendation of Interests and that increased the cache hit ratio in returns.

Our work shows reached the expected results, but we are planning to optimize and increase that outcome also in the research of the future we plan to:

- Apply this approach to a larger topology with a diversity of content traffics for further study.
- This work produces the results of the recommended approach with the LRU, other replacement and placement strategies also will be tested with this approach.
- Concerning the large association rules that may cause congestion in the network; a priority of interests can be set, detect the recommendation interests from other interests.

References

[1] totaluptime, "totaluptime," [Online]. Available: totaluptime.com.

[2] C. A. Kerrche et al., "Emerging Technologies for Connected Internet of Vehicles and Intelligent Transportation System," *Springer Nature Switzerland AG,* 2020.

[3] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine,* vol. 50, pp. 26-36, 2012.

[4] V. Jacobson, M. Mosko, D. Smetters and J. Garcia-Luna-Aceves, "Content-centric networking," *Whitepaper, Palo Alto Research Center,* p. 2–4, 2007.

[5] A. Tariq, R. A. Rehman and B.-S. Kim, "Forwarding Strategies in NDN-Based Wireless Networks: A Survey," *IEEE Communications Surveys Tutorials,* vol. 22, pp. 68-95, 2020.

[6] D. Saxena, V. Raychoudhury, N. Suri, C. Becker and J. Cao, "Named Data Networking: A survey," *Computer Science Review,* pp. 4-5, 2016.

[7] D. Kim and J. Lee, "An NDN Cache Management for MEC," *Appl. Sci.,* p. 2, 2020.

[8] H. Yuan, T. Song and P. Crowley, "Scalable NDN Forwarding: Concepts, Issues and Principles," in *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, 2012.

[9] P. Gasti, G. Tsudik, E. Uzun and L. Zhang, "DoS and DDoS in named data networking," p. 1–7, 2013.

[10] J. Pfender, A. Valera and W. K. G. Seah, "Performance Comparison of Caching Strategies for Information-Centric IoT," in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, New York, NY, USA, 2018.

[11] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs and R. L. Braynard, "Networking Named Content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, New York, NY, USA, 2009.

[12] S. Naz, R. N. B. Rais and A. Qayyum, "A Resource Efficient Multi-Dimensional Cache Management Strategy in Content Centric Networks," *Journal of Computational and Theoretical Nanoscience,* vol. 15, p. 1137–1152, 2018.

[13] S. Ostrovskaya, O. Surnin, R. Hussain, S. H. Bouk, J. Lee, N. Mehran, S. H. Ahmed and A. Benslimane, "Towards Multi-metric Cache Replacement Policies in Vehicular Named Data Networks," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018.

[14] K. N. Lal and A. Kumar, "A Cache Content Replacement Scheme for Information Centric Network," *Procedia Computer Science,* pp. 73-81, 2016.

[15] https://askdatascience.com/. [Online]. Available: https://askdatascience.com/.

[16] "sas," [Online]. Available: https://www.sas.com/en_us/insights/analytics/machine-learning.html.

[17] S. Yagcioglu, "Classical Examples of Supervised vs. Unsupervised Learning in Machine Learning," [Online]. Available: https://www.springboard.com/blog/ai-machine-learning/lp-machine-learning-unsupervised-learning-supervised-learning/.

[18] D. L. Poole and A. K. Mackworth, Artificial Intelligence: Foundations of Computational Agents, 2nd ed., USA: Cambridge University Press, 2017.

[19] A. Sadeghi, G. Wang and G. B. Giannakis, "Deep Reinforcement Learning for Adaptive Caching in Hierarchical Content Delivery Networks," *IEEE Transactions on Cognitive Communications and Networking,* vol. 5, pp. 1024-1033, 2019.

[20] Han, Jiawei, K. Micheline and P. Jian, "Data mining concepts and techniques third edition," *The Morgan Kaufmann Series in Data Management Systems,* 2011.

[21] P.-N. Tan, M. Steinbach, A. Karpatne and V. Kumar, Introduction to Data Mining (2nd Edition), 2nd ed., Pearson, 2018.

[22] Almatten, "Almatten," Almatten, 2021. [Online]. Available: https://www.almatten.com. [Accessed 2021].

[23] "Weka Software," [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/.

[24] nsnam, "NS-3," [Online]. Available: https://www.nsnam.org/.

[25] S. Mastorakis, A. Afanasyev, I. Moiseenko and L. Zhang, "ndnSIM 2: An updated NDN simulator for NS-3," *NDN, Technical Report NDN-0028, Revision 2,* 2016.