Name Popularity Algorithm: A Cache Replacement Strategy for NDN Networks

Andrey Silva, Ivanes Araujo, Neiva Linder and Aldebaro Klautau

***Abstract*—An essential aspect of the network paradigm called Named Data Networking (NDN) is that all nodes in the network have the ability to cache contents natively. The focus of this work relies on efficiently choosing which content should be removed from a full cache to allow storing another one. We propose a new caching replacement algorithm for NDN networks called Name Popularity Algorithm (NPA). The proposed algorithm adopts an extra table that takes into account a long time history for the popularity of contents. Thus, when a data content is removed from the cache, its popularity is not reset, improving the cache usage of the network routers. The performance of NPA is compared to well-known strategies in the literature through NS-3/ndnSIM simulations. The results indicate that the proposed algorithm overcomes others with lower end-to-end delay, higher cache hit ratio and smaller hop count. Besides, our code is available as open source to be accessible for other researchers and to allow reproduction of the results.**

***Index Terms*—NDN, Cache replacement, ndnSIM, ns-3.**

I. Introduction

DUE to the growing demand for multimedia content, forecasts indicate that approximately 78% of the traffic in mobile networks will come from video applications by 2021, while in 2016 it already represented 60% of mobile traffic [1].

To deal with this significant amount of media content it is essential to create distributed caches as close as possible to end-users [2]. This strategy helps to maintain traffic near the access network while reducing the load on the core network and improving the user experience since contents are delivered more quickly and efficiently. Examples of current solutions based on the cache are the Content Delivery Networks (CDNs), which are a set of distributed servers that deliver content copies to end-users [3].

With the Internet evolution, new demands for the network emerged, such as support for scalable distribution of content, mobility and security, among others [4]. These demands have motivated the scientific community to investigate alternative architectures for the Internet of the future. In this context, a new emerging paradigm known as Information-Centric Networking (ICN) has emerged as a promising alternative. ICN has distributed cache as its vital feature and it is expected to improve the network performance by making content retrieval based on names rather than by establishing an end-to-end connection based on an IP address [5]. Hence, ICN enables content caching on several intermediate network nodes (routers for example). Therefore, the content can be easily identified and sent to different users without the need to establish a contact with a server (sometimes located far from the edge of the network where the user is). Thus, this behavior can help to reduce the latency and the chances of network bottlenecks.

A significant amount of research was dedicated to ICN, which led to a wide range of architectures based on ICN principles. Among them, we highlight the main ICN architectures according to [4]: Content Mediator Architecture for Content-Aware Networks (COMET), Scalable and Adaptive Internet Solutions (SAIL), Publish-Subscribe Internet Technology (PURSUIT), Content-Centric Networking (CCN), Data-Oriented Network Architecture (DONA), CONVERGENCE, MobilityFirst and NDN.

Among the architectures cited above, there are two which are prevailing among the competitors: CCN and NDN [6] [7]. Both architectures are leading the on-going effort in ICN standardization process being developed by the Internet research task force (IRTF) information-centric networking research group (ICNRG) [8]. NDN and CCN share the same high-level design and their differences are restricted to specific aspects such as packet format. Therefore, the comparison against the competitors applied to NDN can be also applied to CCN. The NDN/CCN architectures adopt hierarchical names to identify the chunks (pieces of contents). In opposite, DONA and MobilityFirst use flat-label IDs. In CONVERGENCE, PURSUIT and SAIL, the entire content names, or part of them, can be either hierarchical or flat.

Regarding the caching mechanism, all strategies use on-path caching at content routers (the type of caching mechanism can be deterministic or probabilistic). Off-path caching requires additional routing information for NDN/CCN and/or additional registration for MobilityFirst, COMET, SAIL, PURSUIT and DONA. Off-path in COMET relies on unspecified name resolution system.

Although these architectures are different, they all share the following common features: request and response model, content location independence, content-oriented security model and native cache [2]. The focus of this work relies on the NDN architecture. Therefore, the reader interested in more details about the others ICN architecures (e.g., DONA, SAIL, PURSUIT, MobilityFirst, CONVERGENCE, COMET and CCN) is referred to [4].

Similar to other ICN architectures, the NDN architecture is mostly characterized by the exchange of two types of packets: Interest and Data. Both carry a name that identifies a content. In this manner, to retrieve content, a user will send

Andrey Silva, Ivanes Araujo and Aldebaro Klautau are with Federal University of Pará. Belém-PA, Brazil. emails: andreysilva@ufpa.br, aldebaro@ufpa.br, ivanes@ufpa.br.

Neiva Linder is with Ericsson Research, Kista, Sweden. email: neiva.linder@ericsson.com.

Digital Object Identifier: 10.14209/jcis.2019.22

an interest packet to the network with the name of the desired content. The routers will transmit it until it finds a node that contains the requested content, or until it reaches the node that produces it. The node (intermediate or producer) will return the requested data following the reverse path of the interest packet until it reaches the user.

Each NDN node consists of three main structures: Content Store (CS), Forwarding Information Base (FIB) and Pending Interest Table (PIT). The FIB structure can be defined as the NDN forwarding table, responsible for maintaining the prefix of a content name and the next node to which an interest packet should be forwarded. When receiving an interest packet, if a node has the corresponding content in the CS, it sends the data packet by the incoming interface of the interest. Otherwise, it is forwarded to an interface according to the FIB. This same interface is expected to receive the data packet, which is inserted in the CS and forwarded to the interest incoming interface.

The PIT structure has mostly the role of keeping track of this incoming interface [5]. Hence, the data can be delivered to those who requested it by the interest reverse path. In general, the PIT entries are deleted as the data is delivered to the requester or when the time of an interest packet in the PIT expires. Also, the PIT contains recently satisfied interests for loop detection and measurements [9].

Given the relevance of using cache and given that it is native to ICN networks, it is crucial to conduct studies to optimize the usage of cache. This usage includes performing useful strategies to decide which content should be cached and also (the main objective of this work) which content should remain in the cache.

To evaluate the performance of the proposed algorithm, we conducted simulations through NS-3/ndnSIM network simulator comparing our strategy with well-known cache replacement strategies using real network topologies. Each network topology has a variable number of consumers requesting contents from a single server, based on a predefined distribution function. The routers have a limited capacity to store incoming contents in the cache, and they can retrieve the contents when interest packets request them. When the cache is full, the replacement strategy should decide which content remove to store the new incoming content. The obtained results show that our algorithm outperforms the competitors, achieving the highest cache hit ratio, the lowest end-to-end delay, and the lowest hop count. The other competing approaches fail to efficiently handle the router's cache, since they are not using the popularity history of the contents, or they are using deterministic parameters to select which content to remove from the cache. Therefore, our algorithm is more prepared to deal with the nature of the requests.

The main contributions of this paper are listed below.

- We successfully proposed a new strategy for cache replacement in NDN networks, aiming at higher values of the cache hit ratio and the reduction of content delivery time through simulations using real network topologies to validate our findings.
- The proposed strategy provides the ability to efficiently adapt to the content request behavior by using an extension of the CS to store the popularity of the contents discarded from the cache. The algorithm follows a simple math model, improving the router's decision regarding which content should be withdraw from the cache. This approach is intended to ensure the most popular contents will not be lost their popularity, even in cases where they were recently removed from CS (in case of small CS capacity). Therefore, the most requested contents should remain in the cache.
- The proposed algorithm can store the content popularity history with no extra memory cost, depending only on the size of the CS. The algorithm transfers a fixed amount of CS storage capacity (reducing the size of the CS) to the extension table to store contents popularity without losing cache performance. Through simulations, we selected the exact value of capacity transferred from CS, observing the size of the extension table that our algorithm obtained the best performance.
- There are several replacement strategies in the literature. However, it is challenging to find the source code for them. Thereby, a significant number of strategies are difficult to be reproduced. Therefore, we provided our source code as open source[1] to help other researchers to reproduce our results.

The rest of this work is organized as follows: Section II overviews the related work. Section III details the operation of the proposed algorithm. Section IV provides the evaluation of the results obtained through simulation, and Section V concludes the work.

II. Related Work

As in [10], [11], a distinction will be made between *content insertion* strategies and *content eviction* (or *replacement*) strategies. More specifically, it is assumed in this work that the inclusion is done by the Leave a Copy Everywhere (LCE) strategy that is natively used by the NDN architecture [12], simply by including in the CS every incoming data packet. The target then is the replacement strategy.

Whenever a node has reached its maximum storage capacity and needs to insert new content, it follows its replacement strategy and decides which content should be removed from the memory to store the new one. There is a large body of work in this area (see [13] [14]). Some of these methods are based on the *frequency* (or *popularity*) p_c of each content c. Having p_c defined, the replacement decision in this group of strategies is just to discard the content with the smallest p_c. The challenge is then to find a suitable definition and algorithm for p_c. Among these methods, the more widespread and basic is Least Frequently Used (LFU), which adopts p_c as the number of requests for c during the time that the content was stored in the CS.

This method, however, is not robust to significant differences between p_c values (*outliers*), since a given content c can remain in the CS even if no longer requested due to a very large p_c value obtained in the past. This problem is known as "cache pollution," and it is triggered by the fact that the p_c

[1] https://github.com/AndreySilva/NPA-Cache

value in LFU never decreases over time [15]. More advanced versions of the LFU seek to address this issue, and here we will describe two representative strategies.

The LFU-Aging [16] avoids the "cache pollution" by adopting an empirical threshold τ. If the average of p_c values exceeds τ, p_c is changed to $\frac{p_c}{2}$. However, this strategy performance is strictly linked to the adopted τ and there is no robust algorithm for its estimation in distinct networks and traffic conditions.

A variant of LFU-Aging is Least Frequently Used With Dynamic Aging (LFU-DA) [17], which dynamically adjusts and sum the factor γ to p_c value. The value of γ starts at zero and is updated with the value of the content that is removed from the cache. Thus, greater γ values are assigned to contents that are constantly requested, or newly inserted.

The authors in [18] designed the cache policy based on content popularity (CCP), which makes use of a new table called Content Popularity Table (CPT) that stores the cache hits and content popularity to calculate the current content popularity at every end of counting cycles. The authors in [19] proposed the Popularity-based cache policy for Privacy-preserving in NDN (PPNDN) using a probabilistic way based on the content popularity with information of the router. The Least Value First (LVF) algorithm [20] is a replacement algorithm that is based on the value of the delay, age and popularity of the content, which are combined with tuning parameters α, β and ϵ, allowing a priority customization based on the user-provided parameters.

In contrast to popularity-based strategies, there are replacement methods that do not depend on p_c. For example, the algorithm LRU is considered the simplest and most common one and consists of removing the less recently requested content. Two other common examples are: the algorithm RANDOM which randomly chooses a content to be removed from the CS and the First-In-First-Out (FIFO) method, which models the problem as a queue. If the requested content is not in the CS, it is placed at the end of the queue, and all others are moved up one position. In this manner, the content that was at the beginning of the queue (the oldest one in cache) is removed.

Another parameter that can be taken into account for cache replacement decisions is the content packet size. The algorithm SIZE [21], for example, removes from the cache the larger size content. However, a small size content is not always one of the most requested contents. Therefore, a variant of SIZE called Greedy-Dual-Size (GDS) [21] has been proposed, accounting for time, location, size, and other cost information by applying a cost/size function for each block.

By reviewing the previously cited methods, we can infer that the reactive behavior of the algorithms without any knowledge of the recently removed contents or the adoption of adjusting parameters (α, γ, etc.) are the major drawbacks suffered by all proposed algorithms. That is why the newly proposed cache schemes have to deal with these challenges by adopting a mechanism for taking in account the popularity history without the need of using deterministic parameters (which may be suitable only for specific situations) to improve the cache hit ratio and the end-to-end delay. With this motivation, we proposed the NPA algorithm, which can be categorized within the group of replacement strategies that are based on popularity p_c value. It adopts an extra table that allows the algorithm to take into account a long time history for the popularity of contents without any extra deterministic parameter. This way, different of the previous related algorithms, when a content is removed from the cache, the proposed algorithm keeps for a while the popularity of the removed contents. Besides, table extension has been adopted in another works such as [22], [23] to decide which content to store in cache, i.e., for placement strategies. In contrast, the proposed algorithm is a replacement algorithm and is also distinguished by other features as it will be described next section.

III. Proposed Strategy: NPA

In this section, the NPA operation is detailed. Before it can be described, the notation used to represent popularity-based cache strategies is presented in Section III-A. Next, a thorough NPA explanation is made in Section III-B, including the motivation for the NPA features and a brief discussion of its computational complexity.

A. Adopted notation

We will define the notation for popularity-based replacements using, for convenience, the p_c evolution for the LFU strategy. This notation will then be used for explaining the NPA in Section III-B.

M is the CS size. For simplicity, when describing the algorithm, all content packets have the same size.

It is assumed that interests arrive at discrete instants of time denoted by $n \in \mathbb{N}$. Without loss of generality, it is assumed that there is always one, and only one, event related to the cache for each n. For a specific content c, some related events are the *arrival* of an interest for c, the *insertion* of c in the cache and its *eviction*, denoted respectively by a_c, i_c and e_c. The sequence $p_c[n]$ denotes the popularity of a specific content c over time n. The eviction of c from the cache at instant n is denoted here as $p_c[n] = 0$, such that $p_c[n] > 0$ indicates content c is cached at time n. But in LFU implementation, this counter and its associated contents are discarded.

An example using LFU helps grasping the notation. Assume the sequence $\mathbf{E}[n]$ of events $c \in \{A, B, C\}$ is:

$$\begin{aligned}\mathbf{E}[n] = [&a_B, i_B, a_A, i_A, a_B, a_B, a_C, e_A, i_C, a_C,\\ &a_C, a_C, a_A, e_B, i_A, a_B, e_A, i_B]\end{aligned} \tag{1}$$

for $n = [1, 2, \ldots, 18]$, respectively. Assume also that the CS can store $M = 2$ contents. The CS was empty until B arrives, which triggers the first cache inclusion i_B at $n = 2$. Table I depicts the contents in the CS and $p_c[n]$ values used by LFU for different time instants.

It can be noted that due to the insertion strategy adopted, when a content that is not in the cache arrives, for example, $c = C$ as indicated by a_C at $n = 7$, its insertion i_C forces the eviction e_A of the content with smaller $p_c[n]$ (in $n = 8$), which in this case is A. Evictions also happen at instants $n = 14$ and $n = 17$.

For presenting the proposed NPA algorithm, it is useful to define a "relative" time value k_c, which is associated with a specific content c and it is reset to zero whenever c is evicted from a table (such as the CS cache or the additional table used by NPA). Again, sequences indexed by k_c are identified with an over line to avoid ambiguity, and the subscript c is omitted whenever redundant. Therefore, besides $p_c[n]$, the time evolution of a popularity can be alternatively described as $\overline{p}_c[k]$. To avoid confusion, Table II provides an example using the events related to $c = B$ in sequence $\mathbf{E}[n]$ of Table I. Note that, upon eviction, $k = 0$ and its associated sequence also resets with $p_c[0] = 0$ in Table II.

Table I: CS Popularity Evolution for LFU strategy.

Time	Content Name	$\mathbf{p_c^{lfu}[n]}$
$n = 1$	–	–
	–	–
$n = 2$	B	1
	–	–
$n = 3$	B	1
	–	–
$n = 4$	A	1
	B	1
$n = 5$	B	2
	A	1
$n = 6$	B	3
	A	1
$n = 7$	B	3
	A	1
$n = 8$	B	3
	–	–
$n = 9$	B	3
	C	1
$n = 10$	B	3
	C	2
$n = 11$	C	3
	B	3
$n = 12$	C	4
	B	3
$n = 13$	C	4
	B	3
$n = 14$	C	4
	–	–
$n = 15$	C	4
	A	1
$n = 16$	C	4
	A	1
$n = 17$	C	4
	–	–
$n = 18$	C	4
	B	1

Table II: Evolution of the LFU popularity of content B in $\mathbf{E}[n]$ of Table I according to the two kinds of sequences adopted in this work.

$\mathbf{n}$	$\mathbf{p_B^{lfu}[n]}$	$\mathbf{k}$	$\mathbf{\overline{p}_B^{lfu}[k]}$
1	0	0	0
2	1	1	1
5	2	2	2
6	3	3	3
14	0	0	0
16	0	0	0
18	1	1	1

It can be noted from Table II, that the evolution of $\overline{p}_B[k]$ in LFU can be interpreted as an *arithmetic progression* (AP)

$$\overline{p}_B[k] = \overline{p}_B[0] + kr, \tag{2}$$

with a *common difference* $r = 1$. This AP is reset each time content c exits the cache as conveniently indicated by the values of k. The first element $\overline{p}_B[0]$ of Eq. (2) is zero in Table II, but other values can be adopted when Eq. (2) is used for describing NPA in the next section.

B. The Proposed NPA Algorithm

As mentioned, the proposed NPA algorithm adopts an extra table to store and "remember" the popularity of contents even after they are evicted from the cache.

The traditional CS has two fields: the content with S_c bits and the content name hash [24]. The size of CS in LFU is denoted as $M = M^{\text{lfu}}$ to make a distinction between the cache size of other methods. When LFU is adopted, the CS is complemented by an array with M^{lfu} integer values consisting of the LFU popularity $p_c^{\text{lfu}}[n]$.

In NPA, this array is substituted by a table called *history table* (HT), which has space for Y entries with four fields each: a content name hash, a new popularity denoted as $p_c^{\text{npa}}[n]$, and two other integer values: continuous *cache hit* $h_c[n]$ and the *LFU rank* $l_c[n]$. The size of the CS when NPA is adopted is denoted as $M = M^{\text{npa}}$.

Note that the HT and CS tables have distinct sizes Y and M^{npa}, respectively, and the hash field in the HT allows to relate their entries. This way, a content c may have its statistics $(h_c[n], l_c[n], p_c^{\text{npa}}[n])$ stored in HT and not have the corresponding content c cached in CS. Since the space for storing the S_c bits of the content itself in the CS is in average much larger than the one required by the other CS and HT fields [22], [23], it is relatively easy to dedicate some storage space to HT and guarantee that $Y > M^{\text{npa}}$, which is assumed here. A fundamental principle of NPA is then to balance the investment in memory to store contents with what is needed for an auxiliary data structure HT that allows better overall performance.

The value of the LFU rank $l_c[n]$ is similar to that used by LFU when accounting its $p_c[n]$: $l_c[n]$ is incremented for each new interest while the content is in the HT. The distinctions come from the fact that the NPA does not discard the value of $l_c[n]$ when the content leaves the CS, but only when it leaves the HT. Therefore, if a content that remained in the HT returns to the cache, the algorithm can take advantage of this information.

Moreover, if it is necessary to remove an input from HT, it will be removed the one with the lowest $l_c[n]$ value. As defined for $p_c[n]$ in LFU, $l_c[n] = 0$ denotes that content c is not in the HT at instant n.

The counter $h_c[n]$, although stored in the HT, varies solely according to the contents in the CS, i.e., it is reset to zero every time the content is evicted from the CS.

In addition to decreasing the number of times a content's popularity is reset to zero, NPA also adopts the principle of incrementing its popularity $p_c^{\text{npa}}[n]$ more aggressively than the LFU and its variants. Another distinctive feature is that *interest retransmission* is not processed by NPA.

When the interest is not satisfied, it is retransmitted using the same or a different interface. This way, the interests originating from retransmission must be ignored. Ignoring retransmissions is motivated by the fact that too large $p_c^{\text{npa}}[n]$ values could be obtained not due to the popularity of content but due to constant retransmissions of this content in a network experiencing congestion, for example. Therefore, in NPA, if an interest arrives for content c that is already in both CS and HT, and it is not a retransmission, the value of $p_c^{\text{npa}}[n]$ is updated to $p_c^{\text{npa}}[n+1] = p_c^{\text{npa}}[n] + h_c[n]$ and grows faster than in LFU as detailed in the sequel.

Considering $l_c[n]$ influence, $p_c^{\text{npa}}[k]$ evolution for NPA can be written as:

$$p_c^{\text{npa}}[k] = S(k) + l_c[0] - 1 \tag{3}$$

To understand the parcel $l_c[0]-1$, it should be noted that $k = 0$ represents the first interest arrival for c, when it is already in the CS. However, as shown in line 4 from the Algorithm 2, l_c is added to $p_c[k]$ before $k = 0$. Thus, since l_c is incremented by one in $k = 0$ (line 8 from Algorithm 1), the expected influence from l_c in time k should be accounted as $l_c[0] - 1$.

The integers $l_c[n]$ and $h_c[n]$ are essential elements of NPA. To model their operation, we will first define $h_c[n]$ influence in NPA by using the relative time k as:

$$S(k) = (2h_c[0] + k)\frac{k+1}{2} \tag{4}$$

where $S(k)$ represents the sum of the terms of an AP (accounted in line 11 in Algorithm 1), and $h_c[0]$ represents the first therm of the AP summed in $S(k)$, which is typically 1. But it can be a higher value, due to the possibility of having multiple interest arrivals before the content c is cached, making h_c to be incremented each time (line 9 in Algorithm 1). The complete operation of NPA is detailed in Algorithm 1.

Algorithm 1 Proposed NPA algorithm for interest arrival.

```
1:  At_Interest_Arrival (content c)
2:  if not retransmission(c) then
3:     if l_c[n] = 0 then
4:        Exclude from HT the entry with min(l_c[n]);
5:        Insert c in HT, {c was not in HT}
6:        l_c[n+1] ← 1 and h_c[n+1] ← 1;
7:     else
8:        l_c[n+1] ← l_c[n] + 1; {c is already in HT}
9:        h_c[n+1] ← h_c[n] + 1;
10:       if p_c^npa[n] ≠ 0 then
11:          p_c^npa[n+1] ← p_c^npa[n] + h_c[n];
12:       else if c is not pending then
13:          h_c[n+1] ← 1;
14:       end if
15:    end if
16: end if
```

For HT understanding, Figure 1 depicts the data structure adopted in LFU and NPA, where the former can store M^{lfu} and the latter can store M^{npa} contents.

As previously stated, LFU could be modeled as in Expression 2. Therefore, based on the example using $\mathbf{E}[n]$, the evolution of popularity for content B using LFU is shown in Table II. In the same way, the evolution of popularity for content B using NPA is shown in Table III.

Algorithm 2 Proposed NPA algorithm for data arrival.

```
1: At_Data_Arrival (content c)
2: Exclude from CS the entry with min(p_c^npa[n]);
3: Insert c in the cache;
4: p_c^npa[n+1] ← l_c[n];
```

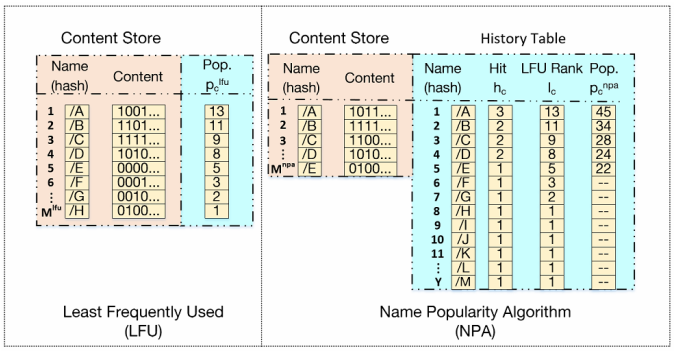


Figure 1: History Table (HT) adopted by the proposed NPA strategy and its comparison with LFU.

Table III: Evolution of the NPA popularity of content B in $\mathbf{E}[n]$ of Table I according to the counters $h_B[n]$ and $l_B[n]$.

n	$\mathbf{p_B^{npa}[n]}$	$\mathbf{h_B[n]}$	$\mathbf{l_B[n]}$
1	0	0	0
2	1	1	1
5	3	2	2
6	6	3	3
14	0	1	3
16	0	1	3
18	**4**	1	**4**

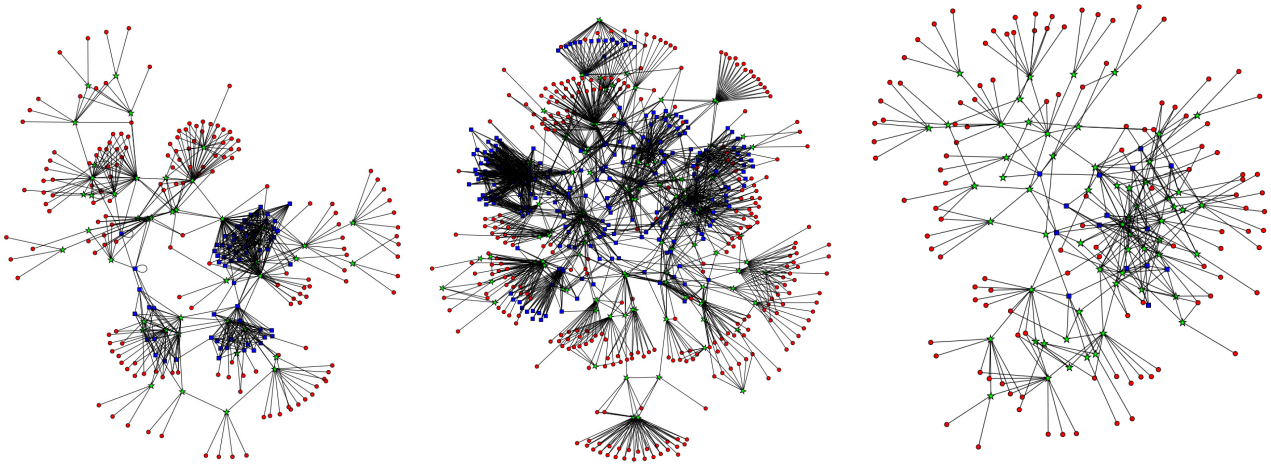
Note that unlike the LFU, the NPA is able to recover the popularity of content B more efficiently, because $p_B^{\text{npa}}[18] = 4$. Since at time $n = 14$ the content B is removed only from the CS, in $n = 18$ the content returns to the cache and NPA can recover part of its popularity with the use of the present value in $l_B[18]$ (which is equal to four), unlike the LFU that has the initial value $p_B^{\text{lfu}}[18] = 1$.

Lastly, since LFU has the computational complexity of $O(\log(n))$, and NPA performs twice the number of insertions, lookups and evictions (because of the extra table), the runtime complexity of NPA is $O(\log(2n))$ which is the same as $O(\log(n))$. Most popular replacement policies range from $O(1)$ to $O(n)$, where n is the number of cached items. But it should be highlighted that the policies that usually achieve good performance have $O(\log(n))$ complexities, like LFU, for example [10].

The next section compares the performance of NPA and the other investigated algorithms, as well as the parameters used in the simulation.

Table IV: Simulation Parameters.

Parameter	Value	Parameter	Value
Simulator	NS-3/ndnSIM 2.2	Zipf α	0.5, 0.6, 0.7, 0.8, 0.9, 1.0
Sim. time	600s	Relative cache size	1%
Num. of seeds	10	Confidence interval	95% (t-distribution)
Request rate	exp(80)	Packet size	4096 Bytes
Num. of producers	1	Routing strategy	Best-route
Contents catalog	10000	Topologies	Telstra, AT&T, Tiscali
Placement strategy	Leave Copy Everywhere (LCE)	Replacement strategy	RANDOM, LFU, LRU, NPA (proposed)



(a) Telstra Topology (b) AT&T Topology (c) Tiscali Topology

Figure 2: Real topologies used in simulations. The red circle, green stars and blue square nodes are consumers, gateway routers and backbone routers, respectively.

IV. Simulation Results

We conducted extensive simulations using the NS-3 discrete simulator through an NDN networking module known as ndnSIM version 2.2. The parameters used in the simulations are listed in Table IV. These parameters were selected based on the previous studies [18] [20] [25] [26]. In total, ten experiments were performed, and a 95% confidence interval was calculated. The popularity of the requested contents follows a Zipf distribution with α parameter ranging from 0.5 to 1.0, where the lower and higher α value represents a low and high correlation between content requests, respectively.

For a fair comparison, it is assumed that the same size in bytes for the LFU cache (and other "baseline" algorithms) must be adopted for NPA to store both CS and its HT.

In this work, it is assumed that all content packets c in CS have the same size and it is adopted $Y = (M^{\text{lfu}} - M^{\text{npa}})C/S$, for Y entries on HT, where C is the content packet size and S is the size of an entry in the HT. From [22] [23], it is adopted $C = 4\ kB$ and $S = 16\ B$.

The simulations were performed using three real network topologies: AT&T, Telstra, and Tiscali (Figure 2). Regarding cache, each node (except the producer and the consumers) can store contents. The storage capacity per node is defined by the relative cache, which is the ratio of the cache size per router to the total number of different contents in the network. The delay and link capacity between backbone routers are 10 ms and 50 Mbps, respectively. From backbone routers to gateway routers the delay is 7 ms with a link capacity of 16.4 Mbps. From consumers to gateway routers, it is adopted a delay of 1.5 ms and ideal links (unlimited capacity).

All topologies have only one producer who is the backbone router with the highest centrality. The number of consumers is not the same for each topology. In this case, the AT&T topology is the one with more consumers.

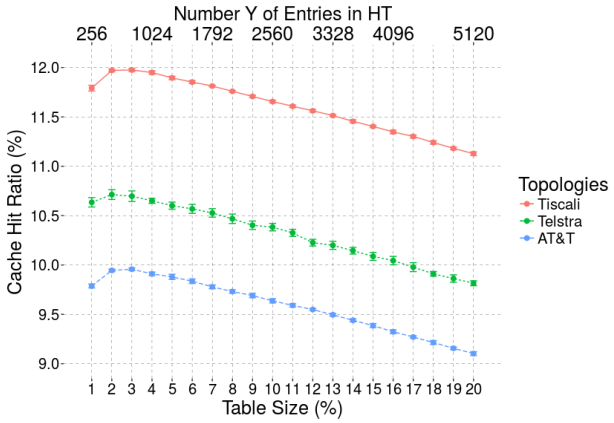
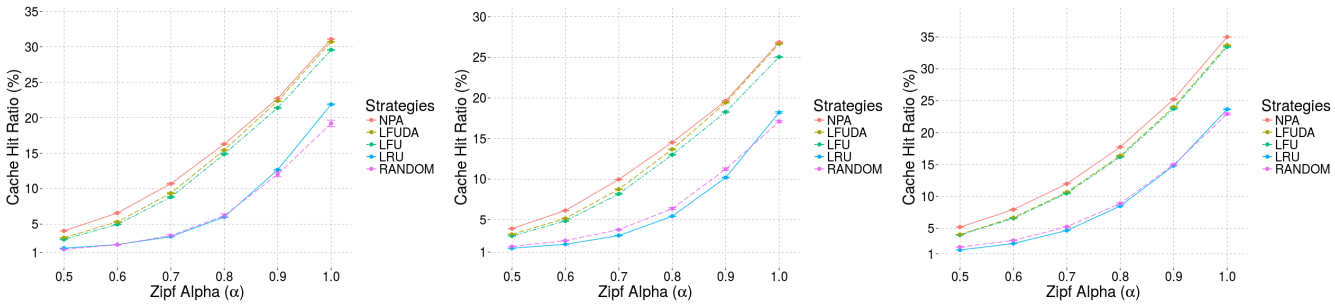


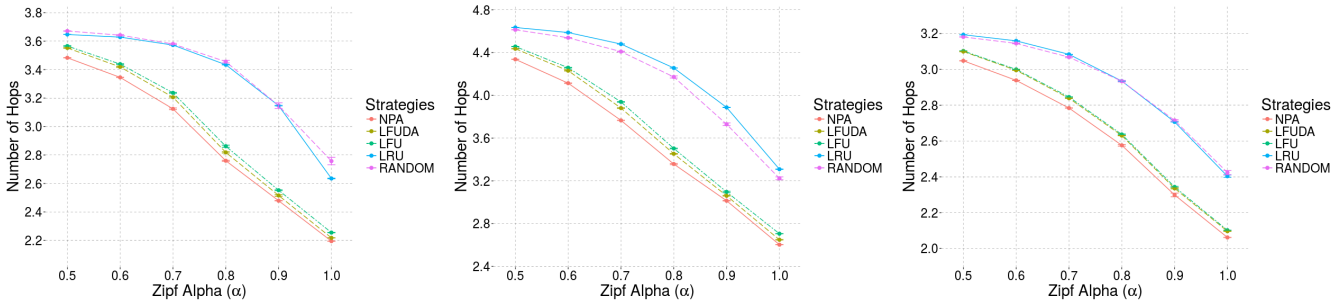
Figure 3: Impact on cache hit ratio with different sizes of HT.

As a first analysis, we intend to observe the size of the HT that obtains the best performance so that this value is subsequently adopted as fixed in all the experiments of this



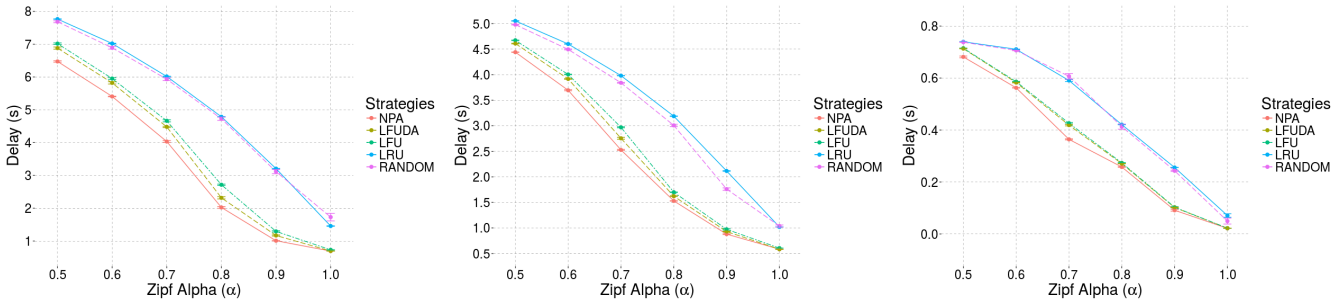
(a) Telstra cache hit ratio (b) AT&T cache hit ratio (c) Tiscali cache hit ratio

Figure 4: Cache hit ratio versus Zipf α parameter.



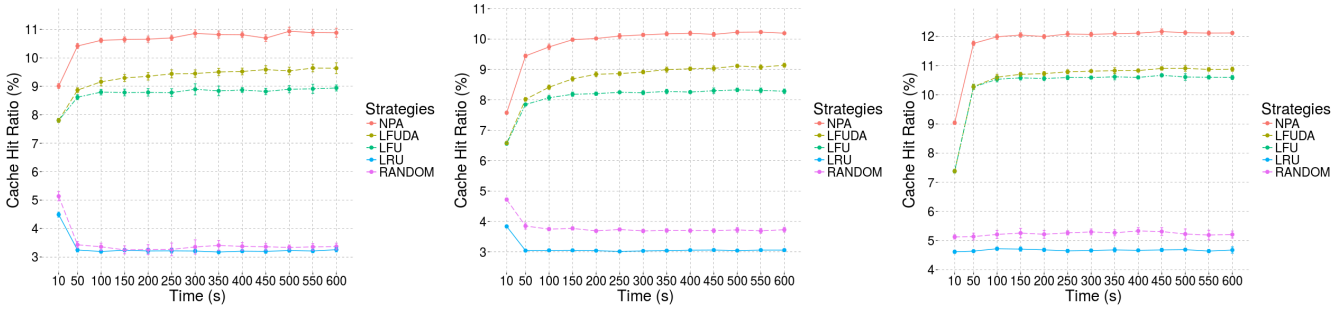
(a) Telstra hop count (b) AT&T hop count (c) Tiscali hop count

Figure 5: Hop count versus Zipf α parameter.



(a) Telstra delay (b) AT&T delay (c) Tiscali delay

Figure 6: End-to-end delay versus Zipf α parameter.



(a) Telstra cache hit ratio over time (b) AT&T cache hit ratio over time (c) Tiscali cache hit ratio over time

Figure 7: Cache hit ratio over time for Zipf $\alpha = 0.7$.

work. The value of the Zipf parameter used for this set of simulations was equal to $\alpha = 0.7$ because it corresponds to a value widely used in the literature and close to that found in real data analysis [27]. Figure 3 shows the impact of the HT size variation in relation to CS size, with the calculated confidence intervals.

The table size percentage from Figure 3 depicts how much of the CS storage capacity has been transferred to HT. For instance, a percentage of 10% indicates that the CS, with a 100 contents capacity (without HT), will be able to store only 90 contents when using the HT and will store 2560 content entries on HT. Thus, the larger the relative size of HT, the smaller the CS one. From Figure 3, we can attest that the HT with 3% of CS size has the best cache hit ratio in all topologies except Telstra. As the HT relative value of 3% was the one that presented the best result in two topologies, this was the value used to generate the comparative results presented below.

Figure 4 presents average cache hit ratio, for all scenarios, with confidence intervals. It can be attested that NPA outperforms the other algorithms (LRU, LFU, LFUDA, and RANDOM) in all scenarios, regardless of the Zipf α and network topology. The two strongest competitors to NPA are LFUDA and LFU, especially when Zipf α is close to 1. NPA good performance for the cache hit ratio can be explained by its ability to register content popularity even after it is removed from the CS. The cache hit ratio gains of NPA concerning LFU have an average of 14.5%, 19.5% and 17.6% for Tiscali, Telstra and AT&T scenarios respectively, and about 12.8%, 13% and 10% concerning LFUDA in the Tiscali, Telstra and AT&T scenarios respectively. The weak cache hit ratio of LRU and RANDOM is because they do not use any popularity information to remove data from the cache. RANDOM results are higher than those of LRU, since the scenarios have a high number of requests, which is very adverse to LRU performance.

Figure 5 illustrates the average hop count necessary to retrieve the content to the user, for all scenarios, with confidence intervals. The hop count metric is related to the distance between the consumer and the requested content. Lower values indicate faster content retrieval to the consumer. As expected, NPA presents better hop count than the other algorithms since it achieves higher cache hit ratio. The hop count reduction of NPA in comparison to LFU is 1.8%, 3% and 3.5% for Tiscali, Telstra, and AT&T scenarios respectively, and about 2%, 1.9% and 2.3% concerning LFUDA in the Tiscali, Telstra and AT&T scenarios respectively.

Figure 6 illustrates the average end-to-end delay, for all scenarios, with confidence intervals. It can be highlighted that NPA achieves a lower end-to-end delay in comparison with the others candidates, which can be explained by the fact that NPA has more cache hits and fewer hops to find the content. This feature also decreases the congestion level on the links, reducing the packet drop/retransmission, which favors faster content delivery. The delay reduction of NPA in relation to LFU is 7.4%, 13.8% and 8.5% for Tiscali, Telstra, and AT&T scenarios respectively, and about 6.5%, 8.2% and 4.8% in comparison to LFUDA in the Tiscali, Telstra and AT&T scenarios respectively. Because of low values of cache hit ratio and higher hop count, LRU and RANDOM provide higher end-to-end delays than other methods.

Figure 7 illustrates cache hit ratio behavior over time using Zipf $\alpha = 0.7$, for all topologies. These results indicate that NPA outperforms the others competitors in all simulation scenarios. The NPA reaches about 12%, 11% and 10% of cache hit ratio in Tiscali, Telstra and AT&T scenarios respectively, attaining at least 12.8% more cache hit ratio than the other depicted strategies.

V. Conclusion

In this work, we have proposed an algorithm (NPA) that uses a table named History Table which increases the capacity to count the content popularity even after it is removed from the cache. Also, a comparative performance study of NPA with other algorithms in the literature was conducted using the NS-3/ndnSIM network simulator in real network topologies.

The proposed algorithm proved to be effective, reaching higher cache hit ratio and lower hop count and end-to-end delay than the other strategies. Therefore, our proposed method can deliver contents faster than compared algorithms, mainly when we analyzed the results for Zipf $\alpha = 0.7$, which is considered to be a realistic value and widely adopted [27]–[29].

As a future work, we aim to expand the algorithm to use an aging parameter to deal with changes in popularity to avoid pollution of cache for contents that are not popular anymore. We intend to compare NPA with more robust cache replacement algorithms, making use of time-varying popularity models (instead of Zipf that generates contents with static popularity). Besides, this paper did not take into consideration the scalability of the proposed algorithm when more than one producer are used. Therefore, we will evaluate the impact of the number of producers on the performance of several cache strategies. Finally, we intend to search for real traces to perform experiments with these data to obtain results closer to real content popularity.

References

[1] C. V. N. Index, "Global mobile data traffic forecast update 2015–2020 white paper. Accessed: Aug. 2, 2016. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.pdf."

[2] L. Saino, "On the design of efficient caching systems," Ph.D. dissertation, University College London, 2015.

[3] H. Yin, X. Zhang, S. Zhao, Y. Luo, C. Tian, and V. Sekar, "Trade-offs between cost and performance for cdn provisioning based on coordinate transformation," *IEEE Transactions on Multimedia*, 2017.

[4] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[5] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[6] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent advances in information-centric networking based internet of things (ICN-IoT)," *IEEE Internet of Things Journal*, 2018.

[7] P. Batista, I. Araújo, N. Linder, K. Laraqui, and A. Klautau, "Testbed for ICN media distribution over LTE radio access networks," *Computer Networks*, vol. 150, pp. 70–80, 2019.

[8] B. Wissingh, C. A. Wood, A. Afanasyev, L. Zhang, D. Oran, and C. Tschudin, "Information-Centric Networking (ICN): CCN and NDN Terminology," Internet Engineering Task Force, Internet-Draft draft-irtf-icnrg-terminology-00, Dec. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-terminology-00

[9] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto *et al.*, "NFD developers guide," Tech. Rep., 2016.

[10] A. Balamash and M. Krunz, "An overview of web caching replacement algorithms," *IEEE Communications Surveys & Tutorials*, vol. 6, no. 2, pp. 44–56, 2004.

[11] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 374–398, 2003.

[12] 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*. ACM, 2009, pp. 1–12.

[13] H. ElAarag, *Web Proxy Cache Replacement Strategies: Simulation, Implementation, and Performance Evaluation*. Springer Science & Business Media, 2012.

[14] A. Ioannou and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2847–2886, 2016.

[15] T. Koskela, J. Heikkonen, and K. Kaski, "Web cache optimization with nonlinear model using object features," *Computer Networks*, vol. 43, no. 6, pp. 805–817, 2003.

[16] M. Arlitt, R. Friedrich, and T. Jin, "Performance evaluation of web proxy cache replacement policies," in *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. Springer, 1998, pp. 193–206.

[17] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, "Evaluating content management techniques for web proxy caches," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 4, pp. 3–11, 2000.

[18] J. hua Ran, N. Lv, D. Zhang, Y. yuan Ma, and Z. yong Xie, "On performance of cache policies in named data networking," in *2013 International Conference on Advanced Computer Science and Electronics Information (ICACSEI 2013)*. Atlantis Press, 2013.

[19] J.-Y. Yang and H.-K. Choi, "PPNDN: Popularity-based caching for privacy preserving in named data networking," in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*. IEEE, 2018, pp. 39–44.

[20] F. M. Al-Turjman, A. E. Al-Fagih, and H. S. Hassanein, "A value-based cache replacement approach for information-centric networks," in *38th Annual IEEE Conference on Local Computer Networks-Workshops*. IEEE, 2013, pp. 874–881.

[21] P. Cao and S. Irani, "Cost-aware www proxy caching algorithms." in *Usenix symposium on internet technologies and systems*, vol. 12, no. 97, 1997, pp. 193–206.

[22] C. Bernardini, T. Silverston, and O. Festor, "MPC: Popularity-based caching strategy for content centric networks," in *2013 IEEE International Conference on Communications (ICC)*. IEEE, 2013, pp. 3619–3623.

[23] M. D. Ong, M. Chen, T. Taleb, X. Wang, and V. Leung, "FGPC: fine-grained popularity-based caching design for content centric networking," in *Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*. ACM, 2014, pp. 295–302.

[24] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto *et al.*, "Nfd developers guide," Technical Report NDN-0021, NDN, Tech. Rep., 2014.

[25] S. Shailendra, S. Sengottuvelan, H. K. Rath, B. Panigrahi, and A. Simha, "Performance evaluation of caching policies in ndn-an icn architecture," in *2016 IEEE Region 10 Conference (TENCON)*. IEEE, 2016, pp. 1117–1121.

[26] Y. Zhang, X. Tan, and W. Li, "PPC: popularity prediction caching in icn," *IEEE Communications Letters*, vol. 22, no. 1, pp. 5–8, 2018.

[27] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 1999, pp. 126–134.

[28] P. Marchetta, J. Llorca, A. M. Tulino, and A. Pescapé, "MC3: A cloud caching strategy for next generation virtual content distribution networks," in *IFIP Networking Conference (IFIP Networking) and Workshops, 2016*. IEEE, 2016, pp. 332–340.

[29] L. Saino, I. Psaras, and G. Pavlou, "Icarus: a caching simulator for information centric networking (icn)," in *Proceedings of the 7th International ICST conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, pp. 66–75.

Andrey Silva received the B.Sc. degree in computer engineering, and the M.Sc. degree in electrical engineering from the Federal University of Pará, Belém, Brazil, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree in electrical engineering. In 2018, he was in a visiting scholar program at Michigan Technological University, Houghton, MI, USA. He is the current president of the IEEE Communication Society (ComSoc) student chapter at Federal University of Pará. His current research interests include caching strategies, computer networks, computer vision, vehicular communications, machine learning, and digital signal processing.

Ivanes Araujo is an Experienced Researcher at Ericsson in Sweden. He is currently working as a 3GPP delegate in the specification of radio interface protocols. His areas of interest are mostly related to 5G and the radio resource control protocol, in topics such as carrier aggregation and dual connectivity. Ivanes Araujo also performed research in information centric networking and quality of service for mobile networks, previously to Ericsson. He holds an M.Sc. in electrical engineering from Federal University of Pará (UFPA), Belém, Brazil.

Neiva Linder is a Research Leader for the Network Orchestration and Automation area at Ericsson Research in Sweden. She has research interests on AI based operations applied to mobile networks automation and service assurance. Neiva joined Ericsson in 2011 and has worked in several technology areas applied to fixed and mobile backhaul network architectures, transport solutions for 4G/5G, including Cloud RAN, SDN, NFV, network slicing, etc. She has over 10 years experience in telecommunication. Previously to Ericsson, Neiva was active in the area of signal processing for communication and held a postdoctoral position at the EIT-LTH Faculty of Engineering, Lund University, Sweden. Neiva Linder holds a Ph.D. in electrical engineering with major in telecommunication from the Federal University of Pará (UFPA), Belém, Brazil.

Aldebaro Klautau received the bachelor (Universidade Federal do Pará, UFPA, 1990), M. Sc. (Universidade Federal de Santa Catarina, UFSC, 1993) and Ph. D. degrees (University of California at San Diego, UCSD, 2003) in Electrical Engineering. Since 1996, he has been with UFPA and is now full professor, the ITU-T TIES Focal Point, and directs LASSE. He was a visiting scholar at Stockholm University, UCSD and The University of Texas at Austin. He is a senior member of the IEEE and a researcher of the Brazilian National Council of Scientific and Technological Development (CNPq). He has supervised more than 50 graduate students, published more than 150 papers in peer-reviewed conferences and journals, and has several international patents. His work focuses on machine learning and signal processing for telecommunications and embedded systems.