A Model for Named Data Networking Inspired by Nonlinear Dynamical Systems

António Rodrigues (up200400437@fe.up.pt)

Abstract—**At the time of its inception, the Internet mostly served the purposes of communication between connected end-hosts. Now, at the World Wide Web era, the Internet is immersed in a content-centric paradigm, more concerned about content generation, sharing and access. Recently, a new research trend — Information Centric Networking (ICN) — started advocating for deep modifications on the Internet's network layer, making it content-centric by design, including the widespread use of in-network caching.**

In this paper, we focus on the analysis of cache behavior in a specific ICN architecture — Named Data Networking (NDN) — under different cache algorithms, network topologies and content usage characteristics. To do so, we specify a simple and but modular NDN router model, loosely inspired in non-linear dynamical systems. We implement the specified model in MATLAB®, providing some simulation results with three simple cache algorithms, specifically (1) Least Recently Used (LRU), (2) More Recently Used (MRU) and (3) Random caching.

I. Introduction

Departing from its initial model as a network for host-to-host communications, the Internet started shifting towards a content-centric model with the advent of the World Wide Web in the 1980s. This model persisted, and with increasingly demanding usage requirements, leading to the development of technologies such as Content Delivery Networks (CDNs) and Peer-to-Peer (P2P) networks [1]. These were built around the architecture's edge, due to the so-called 'ossification' [2] of the Internet's core, leading to inefficiencies in terms of latency, bandwidth usage, among others. Given the widespread adoption of the content-centric model, researchers to think about new and clean-slate designs for the Internet's core, in order for it to natively cope with these issues. Among such efforts [3], the research field of Information Centric Networking (ICN) [4] emerged, advocating the deliberate abolition of network locators, replacing of IP addresses with content identifiers and calling for the widespread use of in-network caching, so that content can be easily served from multiple anywhere in the network [5]–[10]. Here we focus on the aspect of in-network caching in one of such clean slate designs, the Named Data Networking (NDN) architecture [6].

In this paper, we focus on the analysis of cache behavior in NDN networks under different cache algorithms, network topologies and content usage characteristics. To do so, we specify a simple and but modular NDN router model, loosely inspired in nonlinear dynamical systems [11]. We implement the specified model in MATLAB®, providing some simulation results with three simple cache algorithms, specifically (1) Least Recently Used (LRU), (2) More Recently Used (MRU) and (3) Random caching. The main contribution of this work consists in the provision of a framework in MATLAB®, which allows for the simulation of NDN network behavior under different topologies, cache algorithms, NDN router characteristics, etc. without the complexity of more elaborated network simulators.

The remainder of this paper is organized as follows. In Section II we provide an overview over the NDN architecture, focusing on the basic operation of its forwarding engine and the way it involves in-network caching. In Section III, we present the overall methodology followed during this work, including an explanation of the considered NDN router model, network topologies to be considered, cache algorithms, etc. In Section IV we present a set of experiments ran over our model implementation, as well as the respective results. Finally, in Section V we draw some pertinent conclusions from the presented work.

II. Named Data Networking (NDN)

In the Named Data Networking (NDN) [6] architecture, clients issue subscriptions for content objects by specifying a hierarchical (URL-like) content name, e.g. `/pdeec/mtsp/2014/`, which is directly used in NDN packets. Destination network locators (e.g. IP addresses) are not used in this case, as NDN routers are able to forward such packets towards appropriate content-holding destinations, solely based on such names. NDN contemplates two fundamental types of packets, 'Interest' and 'Data' packets, used for content subscriptions and publications, respectively. Interest packets are originally released into the network by clients willing to access a particular content, addressing it via its content name, while Data packets carry the content itself.

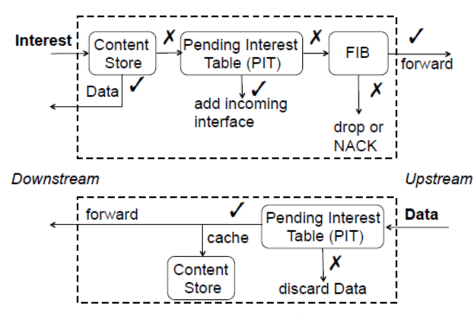


Fig. 1: Interest and Data packet processing according to NDN's forwarding engine [12].

An NDN router is conceptually composed by three main elements: (1) a Forward Information Base (FIB), (2) a Pending Interest Table (PIT) and (3) a Content Store (CS) [6]:

- **Forward Information Base (FIB):** Routing/forwarding table holding entries which relate a name prefix and a list of router interfaces to which Interest packets matching that content name prefix should be forwarded to.
- **Pending Interest Table (PIT):** A table which keeps track of the mapping between arriving Interest packets and the interfaces these have been received from, in order to save a reverse path for Data packets towards one or more subscribers (this may be a 1:N mapping, as an Interest packet matching the same content may be received in multiple interfaces).
- **Content Store (CS):** A cache for content, indexed by content name or item. This novel element allows for content storage at the network level. In-network caching allows an Interest to be satisfied by a matching Data packet in any location other than the original producer of the content, constituting one of the main content-oriented characteristics of NDN.

In NDN, communication is receiver-driven, i.e. having the desire to fetch a particular content, a client releases an Interest packet into the network so that it is forwarded towards an appropriate content holder. In Figure 1 [12], we provide a graphical description of the mechanics of the forwarding engine of an NDN router, supported by the textual description provided below:

1) An Interest packet arrives on an interface (e.g. `iface0`) of an NDN router.
2) A longest prefix match on the content name specified in the Interest (e.g. `name`) is performed. The NDN router will now look in its CS, PIT and FIB, in that order, in order to resume the forwarding action:
    a) If there's a match in the router's CS, a copy of the respective CS entry will be sent back via `iface0`, the Interest packet is dropped. Depending on the pre-specified caching policy (e.g. MRU, LRU, LFU [1], etc.), the organization of the CS may change at this point. **End.**
    b) Else if there is an (exact) match in the PIT, `iface0` is added to the mapping list on the respective entry. The Interest packet is dropped (as a previous one has already been sent upstream). **End.**
    c) Else if only a matching FIB entry is found, the Interest packet is forwarded upstream, via all remaining interfaces on the list (except `iface0`), towards an eventual content holder. A PIT entry <`name`, `iface0`> is added. **End.**
    d) Else if there is no match at all, the Interest packet is simply discarded. **End.**

Note that in NDN only Interest packets are forwarded according to the FIB: intermediate NDN routers (i.e. between client and content holder) forward the Interests and have their respective PIT tables updated with Interest-to-interface mappings, pre-establishing a reverse path for Data packets to follow as soon as a content holder is found. When the reverse path is 'followed' (i.e. in the 'downstream' direction, lower part of Figure 1), each intermediate NDN router receiving a Data packet looks in its PIT for <`name`, `iface`> entries, and forwards the Data packet through all matching interfaces. In addition, a CS entry is created to cache the content locally at the router (again, depending on the caching policy, the organization of the CS may change at this point). If a Data packet with no matching PIT entries arrives, it is treated as unsolicited and discarded.

III. Methodology

In this section we describe the proposed NDN model in detail. We start with an overview, setting the notation and crudely establishing its relation with nonlinear dynamical systems. We then continue with a detailed description of each model component and related procedures.

A. Overview

We consider a conceptual NDN network composed by three main types of entities: (1) $|R|$ NDN routers[2] R_r, $r = \{1, 2, ..., |R|\}$, organized in some type of topology (e.g. cascade, tree, etc.); (2) a set of $|C|$ clients C_c, $c = \{1, 2, ..., |C|\}$; and a single content server S, holding $|O|$ different content objects O_o, $o = \{1, 2, ..., |O|\}$ (e.g. $|O|$ different photos).

Clients issue requests for content objects O_o, i.e. Interest packets i_{O_o}, which are propagated through NDN routers towards the content server S, and eventually followed by Data packets d_{O_o}, containing the requested content object. We represent the elementary set of signals fed to/read from the inputs/outputs of the aforementioned basic entities, at some discrete time n, as a $2\,|O| \times 1$ vector in the form

$$\mathbf{v}[n] = \begin{bmatrix} i_{O_1} \\ i_{O_2} \\ \dots \\ i_{O_{|O|}} \\ d_{O_1} \\ d_{O_2} \\ \dots \\ d_{O_{|O|}} \end{bmatrix} \quad (1)$$

Each component i_{O_o} or d_{O_o} may assume an integer value, i.e. $i_{O_o}, d_{O_o} \in \mathbb{N}_0 = \{0, 1, 2, ...\}$, representing the absence (in case of $i_{O_o}, d_{O_o} = 0$) or presence (in case of $i_{O_o}, d_{O_o} > 0$) of an Interest/Data packet, at a given discrete time n. E.g. considering a setting with $|O| = 2$ content objects, a value of $\mathbf{x}$ corresponding to the presence of two Interests for content O_1 and one Data packet for O_2 (with the absence for the remaining components) would be encoded as

[1] http://en.wikipedia.org/wiki/Cache_algorithms

[2] Here we use the notation $|E|$ to represent the number of elements of type E.

$$\mathbf{v}[n] = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2}$$

With this representation of $\mathbf{v}$, we capture situations in which a network entity may simultaneously receive/issue any Interest or Data packet as its input/output. Our models for NDN routers, clients or servers can be seen as nonlinear dynamical subsystems, which accept inputs $\mathbf{u}$ and produce outputs $\mathbf{y}$, in the form of the vectors shown in 1 and 2. Furthermore, each one of these subsystems exhibits one or more types of state $\mathbf{x}$ (e.g. the composition of a Content Store or a Pending Interest Table), which evolves according to nonlinear dynamics driven by $\mathbf{u}$ and the current state $\mathbf{x}$:

$$\mathbf{x}[n+1] = \mathcal{H}(\mathbf{x}[n], \mathbf{u}[n]) \tag{3}$$

Outputs $\mathbf{y}$ are nonlinearly related to some current state $\mathbf{x}$ and input $\mathbf{u}$:

$$\mathbf{y}[n] = \mathcal{G}(\mathbf{x}[n], \mathbf{u}[n]) \tag{4}$$

We also note that in some subsystems, namely at the clients C, the outputs $\mathbf{y}$ may include a stochastic component w (e.g. in the case that Interest signals are randomly generated):

$$\mathbf{y}[n] = \mathcal{G}(\mathbf{x}[n], \mathbf{u}[n]) + w \tag{5}$$

In the next subsections, we describe each one of the nonlinear subsystems in detail, including the nature of the types of state and their nonlinear dynamics.

B. NDN Router Model

An NDN router is the central entity of the presented model, acting as the main agent of NDN's forwarding engine. It is also the more complex entity, including a set of submodules, already mentioned in Section II: (1) the Forward Information Base (FIB); (2) the Pending Interest Table (PIT); and (3) the Content Store (CS). We first provide an overview of our NDN router module, and then proceed with the description of each one of its submodules.

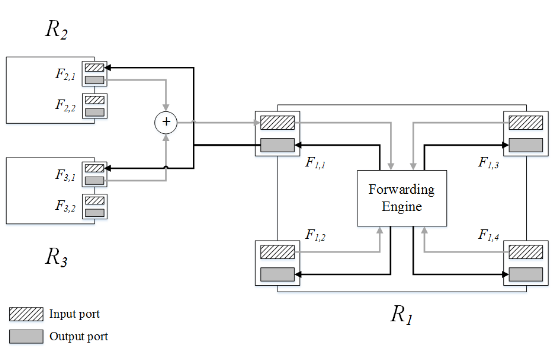


Fig. 2: Graphical depiction of our NDN router model.

Figure 2 provides a graphical description of the proposed router model. A router R contains a set of $|F|$ interfaces F, used to interconnect it to other entities (e.g. some other router R', a client C or the server S). Each interface can subsequently be divided into one input and one output port, which 'cross-connect' with the ports of the attached interfaces (see Figure 2). We use the notation $F_{r,f,in}$ or $F_{r,f,out}$, to refer to the input/output ports of an interface f, of a router R_r.

The act of forwarding some set of Interest/Data packets from a router R_1, over some interface F_1, is modeled by having R_1 fill $F_{1,1,out}$ with some set of signals $\mathbf{y}$, following the encoding shown in Section III-B. Conversely, the act of receiving some set of Interest/Data packets is modeled by having routers R_2 and R_3 — connected with $F_{1,1}$ via $F_{2,1}$ and $F_{3,1}$ — fill $F_{2,1,in}$ and $F_{3,1,in}$ with $\mathbf{u}$ [3]. As seen in Figure 2, more than one entity may be connected to some interface, in which case the signals $\mathbf{y}$ originating from the interconnected interfaces' output ports, e.g. $F_{2,1,out}$ and $F_{3,1,out}$, are combined and summed at the other end's input port, e.g. $F_{1,1,in}$. While the use of interfaces and input/output ports may be seen as a case of over engineering, we argue it makes our model robust, highly modular and capable of supporting multiple network topologies.

1) Pending Interest Table (PIT): In the same way that the FIB commands the forwarding of Interests, the PIT commands the forwarding of Data packets. Nevertheless, the composition of the PIT is more dynamic than that of the FIB, being dependent on the flow of Interest and Data packets through the NDN router. We model the PIT as a $|O| \times |F|$ matrix in the form:

$$\mathbf{PIT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{6}$$

PIT[4] entries (o, f) are encoded as 0 or 1: if $(o, f) = 1$, Data packets of type O_o have been previously requested via interface F_f, and so Data packets d_{O_o} shall be forwarded via F_f; on the other hand, if $(o, f) = 0$, d_{O_o} should not be forwarded via that interface. The arrival of Interest and Data packets influences the composition of the PIT over time, each triggering special routines — **PIT::updateOnInterest()** and **PIT::updateOnData()** — shown below. We consider special $2|O| \times |F|$ matrices $\mathbf{U}$ and $\mathbf{Y}$, corresponding to the concatenation of all the column vectors $\mathbf{u}_{r,f}$ and $\mathbf{y}_{r,f}$, i.e. the contents from the input and output ports of all the interfaces F_f, at some router R_r:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_{r,1} & \mathbf{u}_{r,2} & \dots & \mathbf{u}_{r,|F|} \end{bmatrix} \tag{7}$$

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_{r,1} & \mathbf{y}_{r,2} & \dots & \mathbf{y}_{r,|F|} \end{bmatrix} \tag{8}$$

[3] In fact, this procedure can be extended to any network entity, let it be a router, a client or the server.

[4] We use the form 'PIT' for general references to the Pending Interest Table, and '**PIT**' when referring to its matrix form, as in 6. This dual representation is extended to the FIB and CS.

1: **define** PIT::updateOnInterest($\mathbf{U}$):
2:
3: $\mathbf{U}' \leftarrow \mathbf{U}(1:|O|,:)$
4: $\mathbf{Y}_i' \leftarrow \neg(\mathbf{PIT} \times \mathbf{1}^{|F|})\ \&\ (\mathbf{U}' \times \mathbf{1}^{|F|})$
5: $\mathbf{PIT} \leftarrow \mathbf{PIT}\ |\ \mathbf{U}'$
6: **return** $\mathbf{Y}_i'$

Upon the reception of Interest signals, i.e. $\mathbf{U}'$ (the first $|O|$ rows of $\mathbf{U}$), we first identify the content items O_o, or the row indexes o of the **PIT**, for which there are **no** pending Interests (line 4). For convenience, we often recur to binary operations (negation '$\neg$', conjunction '&', disjunction '|') over matrices: e.g. in line 4, after summing all columns of the **PIT**[5], we negate the result, obtaining a binary encoded $|O| \times 1$ column vector which indicates the absence (encoded as '1') and presence (encoded as '0') of pending Interests for some content object O_o. Still in line 4, we obtain a $|O| \times 1$ column vector, $\mathbf{Y}_i'$, which encodes the Interest signals that the NDN router needs to forward upstream. This is accomplished by taking the logic AND, '&', between $\neg(\mathbf{PIT} \times \mathbf{1}^{|F|})$ and $\mathbf{U}'$. Note that even if $\mathbf{U}'$ includes some $i_{O_o} > 1$, we only need to forward one Interest over the interfaces specified in the FIB, and so the binary encoding of $\mathbf{Y}_i'$, resulting from the use of binary operations, neatly serves our purposes. In line 5, the contents of the **PIT** are updated by performing a logic OR, '|', with $\mathbf{U}'$, so that it registers all the newly received Interest signals and their correspondence to interfaces. This last step is important, as it allows future Data packets to be forwarded downstream over the requesting interfaces.

1: **define** PIT::updateOnData($\mathbf{U}$):
2:
3: $\mathbf{U}' \leftarrow \mathbf{U}(|O|+1 : 2\,|O|,:)$
4: $\mathbf{G} \leftarrow (\mathbf{U}' \times \mathbf{1}^{|F|}) \times \mathbf{1}^{|F|^T}$
5: $\mathbf{Y}_d \leftarrow \mathbf{PIT}\ \&\ \mathbf{G}$
6: $\mathbf{PIT} \leftarrow \mathbf{PIT}\ \&\ \neg\mathbf{G}$
7: **return** $\mathbf{Y}_d$

Note that the objective of PIT::updateOnData() is the reverse operation of PIT::updateOnInterest(), i.e. forwarding Data packets over the interfaces for which there is a registered Interest in the **PIT**, with the result encoded in matrix $\mathbf{Y}_d$. $\mathbf{G}$ consists in a $|O| \times |F|$ matrix, which expands the single-column vector $\mathbf{U}' \times \mathbf{1}^{|F|}$ to $|F|$ columns (line 4), making it ready for the subsequent AND ('&') operations (lines 5 and 6). The final step of the operation (line 6) consists in erasing all Interest registrations which have been successfully attended.

As a final remark, we can now identify the **PIT** as one form of state $\mathbf{x}$, which is driven by the nonlinear dynamics specified on the PIT::updateOnInterest() and PIT::updateOnData() routines (lines 5 and 6, respectively). Furthermore, both routines also contribute to the generation of outputs, in the form of matrices $\mathbf{Y}_d$ and $\mathbf{Y}_i'$.

2) Content Store (CS): We model the Content Store (CS), i.e. the NDN router's cache, as an $|O| \times |P|$ matrix, in which $|P|$ is the size of the CS, i.e. maximum number of content objects it is able to accommodate at any given point in time. We use P to represent a position or slot in the CS, which is able to hold a single content object O_o (here we do not consider a notion of content object size, an object always fits in a slot P). E.g. in 9 we show the encoding of the **CS** for R_1 in Figure 3, with $|P| = 2$, which is shown to contain content objects O_2 and O_4[6].

$$\mathbf{CS} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{9}$$

Again, we follow a binary encoding, using $(o,p) = 1$ to indicate the presence of content O_o at slot P_p, and $(o,p) = 0$ as an indication of its absence. Each slot is assigned a different integer index, i.e. P_p, with $p = \{1, 2, ..., |P|\}$, which express the idea of 'cache levels': O_o, occupying slot P_2, is at the 2nd (highest) position of the **CS**. P_1 is usually interpreted as the 'highest' level, and $P_{|P|}$ the 'lowest', nevertheless the meaning is of the 'levels' is often dependent on the cache algorithm implemented by the CS.

Note that **CS** must obey a set of constraints, since (1) each slot can only hold a single content object, and (2) it is inefficient for the **CS** to hold multiple copies of a content O_o. Specifically, the conditions for the validity of **CS** are:

$$\sum_{o=1}^{|O|} \mathbf{CS}_{o,p} \leq 1 \quad \forall\ p \in 1, 2, ..., |P| \tag{10}$$

$$\sum_{p=1}^{|P|} \mathbf{CS}_{o,p} \leq 1 \quad \forall\ o \in 1, 2, ..., |O| \tag{11}$$

Similarly to the PIT, the operations related to the CS are implemented by two routines: **CS::updateOnInterest()** and **CS::updateOnData()**. While the behavior of CS::updateOnData() is specific to a particular cache algorithm (see Section III-D), that of CS::updateOnInterest() is rather general, consisting in the generation of a pair of $|O| \times |F|$ matrices, $[\mathbf{Y}_h, \mathbf{R}]$. In short, $\mathbf{Y}_h$ encodes the Data packets for which there are cache hits, already assigned to the appropriate interface columns; and $\mathbf{R}$ encodes all the Interest signals for which there were no cache hits[7].

Again, **CS** may be seen as the other form of state in the NDN router subsystem, driven by the nonlinear dynamics specified by each different cache algorithm. Furthermore, the CS::updateOnInterest() routine, common to all caching policies, contributes to the generation of the output matrix $\mathbf{Y}$ via $\mathbf{Y}_h$.

3) Forwarding Information Base (FIB): The FIB is important for the act of forwarding Interest packets towards appropriate content sources, by indicating the interfaces F over which such sources are reachable. For each NDN router R, we model the FIB as a simple $|O| \times |F|$ matrix in the form

[5] We use the notation $\mathbf{1}^m$ for the $m \times 1$ sum vector, i.e. $\mathbf{1}^m = [1\,1\,...\,1]^T$.

[6] Usually, we consider $|P| << |O|$.

[7] Due to space constraints, we do not show the algorithm of CS::updateOnInterest(), which can nevertheless be consulted in github.com/adamiaonr/mtsp-project.

$$\mathbf{FIB} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{12}$$

The **FIB** encoding shown in 12 would be held by router R_1 in the simple topology shown in Figure 3. **FIB** entries (o, f) are encoded as 0 or 1: if $(o, f) = 1$, content objects of type O_o are accessible via interface F_f, meaning that Interests i_{O_o} shall be forwarded via F_f; on the other hand, if $(o, f) = 0$, i_{O_o} should not be forwarded via that interface. Following the same type of operations described for the PIT in Section III-B1, we now present the algorithm followed by NDN routers to forward Interest signals, which uses PIT::updateOnInterest() as well as CS::updateOnInterest():

1: **define** Router::forward($\mathbf{U}$):
2:
3: $[\mathbf{Y}_h, \mathbf{R}] \leftarrow$ CS::updateOnInterest($\mathbf{U}$)
4: $\mathbf{Y}_i' \leftarrow$ PIT::updateOnInterest($\mathbf{R}$)
5: $\mathbf{Y}_i \leftarrow \mathbf{FIB}\ \&\ (\mathbf{Y}_i' \times \mathbf{1}^{|F|^T})$
6: $\mathbf{Y}_d \leftarrow$ PIT::updateOnData($\mathbf{U}$)
7: $F_{n,1:|F|,out} \leftarrow (\mathbf{Y}_i + \mathbf{Y}_h + \mathbf{Y}_d)$

From CS::updateOnInterest(), we obtain $\mathbf{Y}_h$, the output Data signals resulting from cache hits, and $\mathbf{R}$, the Interest signals for which there were no cache hits. We use $\mathbf{R}$ as an input to PIT::updateOnInterest(), ultimately obtaining $\mathbf{Y}_i$, the output Interest signals. Finally, the router processes the Data input signals on $\mathbf{U}$, generating the Data output signals, $\mathbf{Y}_d$, according to the behavior of PIT::updateOnData(). The last operation (line 6) refers to the (parallelized) ‘filling’ of the output ports of all the interfaces F of router R_r, with both $\mathbf{Y}_i$, $\mathbf{Y}_h$ and $\mathbf{Y}_d$, which together consist in the complete output matrix $\mathbf{Y}$.

In our model, the composition of the **FIB** is established at an initial phase, not suffering any further alterations (more details in Section III-F). Note that the FIB only commands Interest forwarding actions, not participating in the forwarding of Data packets.

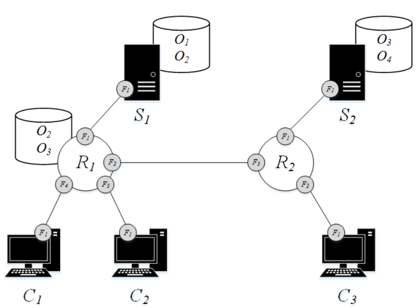


Fig. 3: Simple example of an NDN network topology.

C. Endpoints

In our model we consider two forms of endpoints: clients C and servers S. Clients generate Interests for content, i.e. vectors in the form of $\mathbf{v}$ in 1, while servers hold caches in which content objects persist and are always available.

The operation of servers is simple, basically consisting in mirroring the Interest signals received as input, according to the contents of their persistent CS:

define Server::mirror($\mathbf{U}$):

$\mathbf{U}' \leftarrow \mathbf{CS}\ \&\ \mathbf{U}$

$\mathbf{Y}_h \leftarrow \begin{bmatrix} \mathbf{0}^{|O|} \times \mathbf{1}^{|F|^T} \\ \mathbf{U}'(1 : |O|, :) \end{bmatrix}$

$F_{n,1:|F|,out} \leftarrow \mathbf{Y}_h$

Clients generate Interest signals in a probabilistic fashion, e.g. according to some popularity distribution such as Zipf [13]. In our model, clients C take (...).

D. Cache Algorithms

E. Network Topologies

We represent a network topology using a square matrix $\mathbf{T}$, with dimension $|R| + |C| + |S|$. For representation purposes, we attribute an integer index to each one of the network nodes, and so each elements (i, j) of the matrix corresponds to interconnections between network entities with indexes i and j. The values at each (i, j) position identify the **near end interface** of the connection, i.e. the interface at entity i. E.g. the matrix $\mathbf{T}$ shown in 13 encodes the topology shown in Figure 3. In this case, we attribute the integers 1 to 7 to the network entities, starting with the clients C_1 to C_3, then routers R_1 and R_2 and finally the servers S_1 and S_2.

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 4 & 3 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{13}$$

F. ‘Connecting the Dots’

G. Implementation

The NDN network model described above has been implemented in MATLAB®, taking advantage of its Object Oriented Programming (OOP) capabilities. Most of the object types are independent of particular cache algorithms, nevertheless CS objects are defined as abstract classes, allowing for specific implementations of cache policies such as LRU, MRU and Random caching. The MATLAB® code — along with additional documentation — is freely available at github.com/adamiaonr/mtsp-project.

IV. Experiments

This section describes a set of experiments, conducted to evaluate the behavior of our NDN model under different cache conditions. In Section IV-A, we start with a brief description of the setup of our experiments, including relevant parameters (e.g. number of content objects, popularity distribution, etc.), network topologies, cache characteristics, as well as the metrics to be considered. Later, in Section IV-B, we present the results of several experiment runs, according to the metrics specified in Section IV-A.

A. Experimental Setup

The conducted experiments can be characterized along three main dimensions: (1) content object characteristics; (2) network topologies; and (3) cache characteristics.

1) Content Object Characteristics: For all experiments we consider a content object space of $|O| = 100$, with decreasing popularity and following a Zipf distribution: each content object O_o, with $o = \{1, 2, ..., |O|\}$, is requested by a client C with probability $q_o = \frac{c}{o^{\alpha}}$, with $c = 0.8$ and $\alpha \in \{0.5, 1, 2\}$.

At each experiment round (see Section III-F for more details on rounds), each client C generates a signal for content object O_o with probability q_o. We consider a number of rounds $|N| = 10000$ for all experiment runs.

2) Network Topologies: We consider two types of topologies: (1) a cascade topology with $|L| = 5$ levels ($|C| = 1$, $|R| = 1$ and $|S| = 1$), shown in Figure 4(a); (2) a binary tree topology, also with $|L| = 5$ levels ($|C| = 8$, $|R| = 7$ and $|S| = 1$), as shown in Figure 4(b).

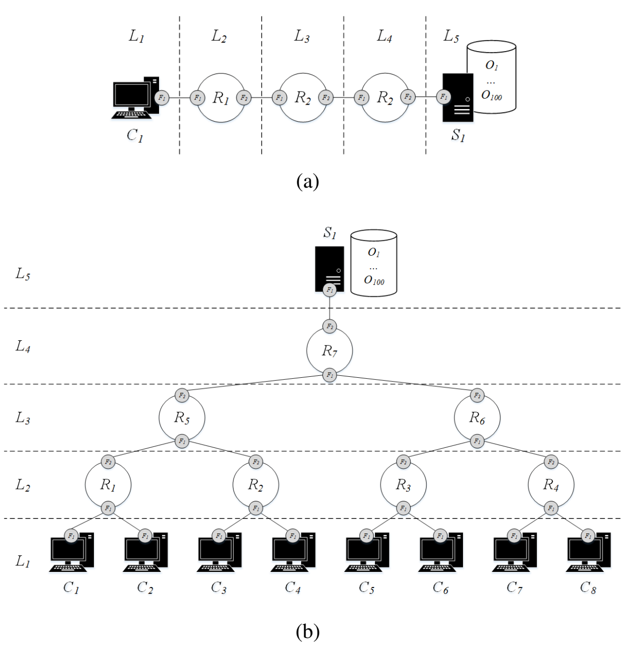


Fig. 4: Graphical depiction of the topologies considered for our experiments: (a) cascade topology ($|L| = 5$, $|C| = 1$, $|R| = 3$ and $|S| = 1$); (b) binary tree topology ($|L| = 5$, $|C| = 8$, $|R| = 7$ and $|S| = 1$).

The definition of 'topology level' is straightforward and depicted in Figure 4. Furthermore, we set the FIBs of every NDN router to forward Interests over the upstream interfaces, i.e. $F_{r,2}$, resulting in the form:

$$\textbf{FIB} = \begin{bmatrix} 0 & 1 & 0 & ... & 0 \\ 0 & 1 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 1 & 0 & ... & 0 \end{bmatrix} \tag{14}$$

3) Cache Characteristics: We evaluate the performance of our model under three different types of cache algorithms, already described in Section III-D: (1) Least Recently Used (LRU); (2) More Recently Used (MRU); and (3) Random caching. We consider different cache sizes, specifically $|P| = \{10, 25, 50, 75\}$.

4) Metrics: For all our experiments we collect and evaluate the following metrics:

1) Total number of Interests and Data packets received/sent per network node/level;
2) Cache hit/miss rate per content object and network level;
3) Ratio of received Interests to original requests, per content object and network level (including server(s) S);
4) Relative time spent at cache per content object.

Regarding metric 2, we first provide our definition of 'cache hit': a cache hit happens when an Interest signal for some content object O_o, arriving at some router R_r, finds a cached copy of O_o at the CS of R_r. Therefore, to compute metric 2, we consider the Interest signals arriving at all NDN routers of some level L, discriminated by content object O_o, for all simulation rounds N. We then find the ratio between Interest 'hits' for O_o, i'_{O_o}, and the total value of arriving Interests for O_o at that level, i_{O_o}:

$$\frac{\sum_{n=1}^{|N|} i'_{O_o}}{\sum_{n=1}^{|N|} i_{O_o}} \quad \forall\ R_r \text{ in level } L \tag{15}$$

For metric 4, we simply find the average number of rounds some content object O_o spends at the caches of NDN routers of some level L, and find the ratio vs. the total number of simulation rounds.

B. Experimental Results

Here we present a selection of experimental results, following the metrics introduced in Section IV-A[8]. The captions on the figures mention the respective experimental parameters.

V. Discussion and Conclusions

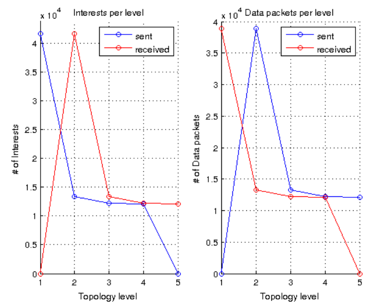
References

[1] Andrea Passarella. A Survey on Content-Centric Technologies for the Current Internet: CDN and P2P solutions. *Computer Communications*, 35(1):1–32, 2012.

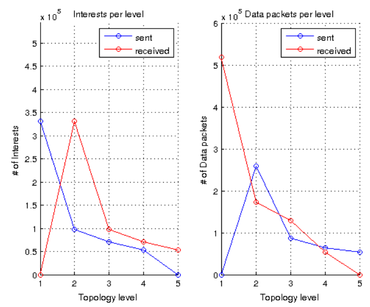
[2] M Handley. Why the Internet Only Just Works. *BT Technology Journal*, 24(3):119–129, 2006.

[3] J Pan, S Paul, and R Jain. A Survey of the Research on Future Internet Architectures. *Communications Magazine, IEEE*, 49(7):26–36, July 2011.

[8]A complete collection of results can be found in github.com/adamiaonr/mtsp-project/tree/master/report/figures/experiments.

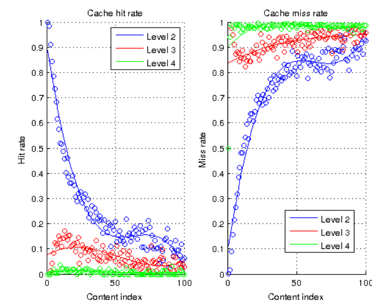


(a)

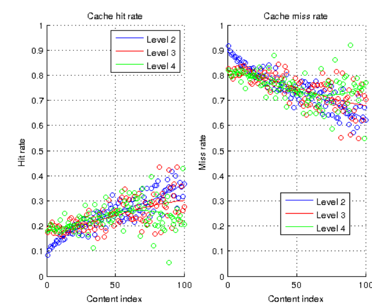


(b)

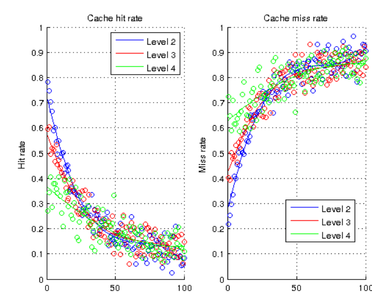
Fig. 5: Number of Interests/Data packets sent/received per topology level, for different topologies: cascade (a), binary tree (b). $|P| = 25$, $\alpha = 1$, LRU caching.



(a)



(b)



(c)

Fig. 6: Cache hit/miss rates for different cache algorithms: LRU (a), MRU (b) and Random caching (c). Cascade topology, $|L| = 5$, $|C| = 8$, $|R| = 7$ and $|S| = 1$, $|P| = 25$, $\alpha = 1$.

[4] George Xylomenos, Christopher N. Ververidis, Vasilios a. Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V. Katsaros, and George C. Polyzos. A Survey of Information-Centric Networking Research. *IEEE Communications Surveys & Tutorials*, PP(99):1–26, 2013.

[5] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A Data-Oriented (and Beyond) Network Architecture. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 181–193, 2007.

[6] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking Named Content. *Proceedings of the 5th international conference on Emerging networking experiments and technologies CoNEXT 09*, 178(1):1–12, 2009.

[7] Dirk Trossen and George Parisis. Designing and Realizing an Information-Centric Internet. *IEEE Communications Magazine*, 50(7):60–67, 2012.

[8] Dipankar Raychaudhuri, Kiran Nagaraja, and Arun Venkataramani. MobilityFirst : A Robust and Trustworthy Mobility- Centric Architecture for the Future Internet. *ACM SIGMOBILE Mobile Computing and Communications Review*, 16(3):2–13, 2012.

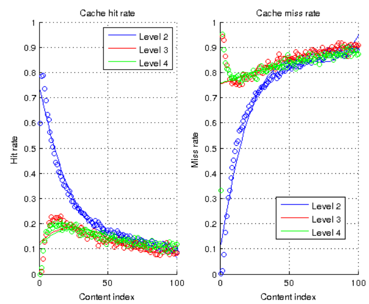
[9] Dongsu Han, Peter Steenkiste, Michel Machado, and David G Andersen. XIA : Efficient Support for Evolvable Internetworking. In *The 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI'12)*, pages 1–14, 2012.

[10] Christian Dannewitz, Dirk Kutscher, BöRje Ohlman, Stephen Farrell, Bengt Ahlgren, and Holger Karl. Network of Information (NetInf) - An Information-Centric Networking Architecture. *Comput. Commun.*, 36(7):721–735, April 2013.

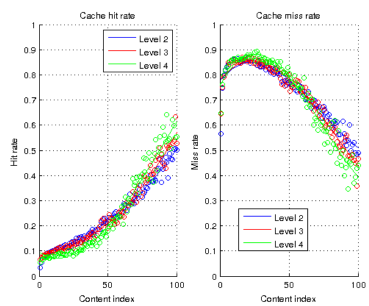
[11] J. K. Hedrick and A. Gira. *Control of Nonlinear Dynamic Systems: Theory and Applications*. 2010.

[12] Cheng Yi, Alexander Afanasyev, Lan Wang, Beichuan Zhang, and Lixia Zhang. Adaptive Forwarding in Named Data Networking. *ACM SIGCOMM Computer Communication Review*, 42(3):62–67, June 2012.

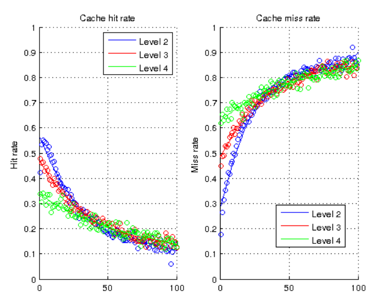
[13] G Carofiglio, M Gallo, L Muscariello, and D Perino. Modeling Data Transfer in Content-Centric Networking. In *Teletraffic Congress (ITC), 2011 23rd International*, pages 111–118, September 2011.



(a)



(b)



(c)

Fig. 7: Cache hit/miss rates for different cache algorithms: LRU (a), MRU (b) and Random caching (c). Binary tree topology, $|L| = 5$, $|C| = 8$, $|R| = 7$ and $|S| = 1$, $|P| = 25$, $\alpha = 1$.