A Model for Named Data Networking Caching Policies 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 caching policies, 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. We implement the specified model in MATLAB, providing some simulation results with X simple caching policies, specifically (...).

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 caching policies, 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 X simple caching policies, specifically (...).

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, caching policies, etc. In Section IV we show details about the implementation of such models in MATLAB. In Section V we present a set of experiments ran over our model implementation, as well as the respective results. Finally, in Section VI 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.

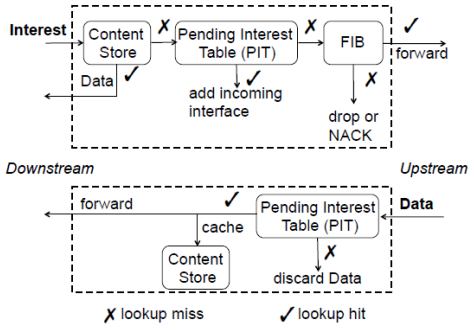

Fig. 1: Interest and Data packet processing according to NDN's forwarding engine [12].

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.

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

III. Methodology

We now present a brief description of the NDN model (...) for this work. We start (...).

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 instant t, as a $2|O| \times 1$ array in the form

$$\mathbf{u}[t] = \begin{bmatrix} i_{O_1} \\ i_{O_2} \\ ... \\ i_{O_{|O|}} \\ d_{O_1} \\ d_{O_2} \\ ... \\ d_{O_{|O|}} \end{bmatrix} \quad (1)$$

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 time instant t. E.g. considering a setting with $|O| = 2$ content objects, a value of $\mathbf{u}$ 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

$$\mathbf{u}[t] = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (2)$$

With this representation of $\mathbf{u}$, we capture situations in which a network entity may simultaneously receive/issue any Interest or Data packet as its input/output.

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

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.

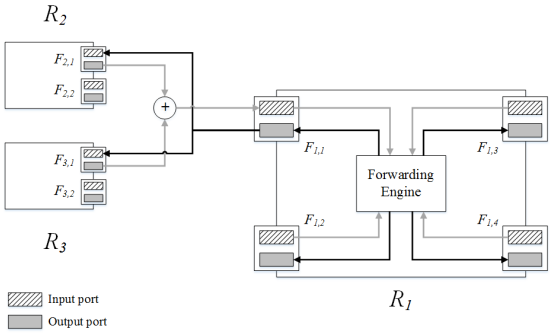

Fig. 2: Graphical depiction of our NDN router model.

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 $\mathbf{u}$, 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 $\mathbf{u}$ 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.

Similarly to the FIB, we model the PIT as a $|O|\times|F|$ matrix in the form

$$\mathbf{PIT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3}$$

PIT[4] entries (o, f) are encoded as 0 or 1: if $(o, f) = 1$, that means that 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. These operations influence the composition of the **PIT** over time, whose maintenance can be summarized by the two simple routines shown below, to be followed upon Interest and Data packet arrivals, respectively. For the algorithms shown below, we consider a special matrix $\mathbf{A}$, corresponding to the concatenation of all the arrays $\mathbf{u}_{r,f,in}$ as columns, i.e. the contents from the input ports of all the interfaces F_f, at some router R_r:

$$\mathbf{A} = \begin{bmatrix} \mathbf{u}_{r,1,in} & \mathbf{u}_{r,2,in} & \dots & \mathbf{u}_{r,|F|,in} \end{bmatrix} \tag{4}$$

1: **define** updateOnInterest($\mathbf{A}$):
2:
3: $\mathbf{A}' \leftarrow \mathbf{A}(1 : |O|, :)$
4: $\mathbf{D} \leftarrow \text{diag}(\neg(\mathbf{PIT} \times \mathbf{1}))$
5: $\mathbf{B} \leftarrow \mathbf{D} \times \mathbf{A}'$
6: $\mathbf{PIT} \leftarrow \mathbf{PIT} \mid \mathbf{A}'$
7: **return B**

Upon the reception of Interest signals, i.e. $\mathbf{A}'$ (the first $|O|$ rows of $\mathbf{A}$), we first identify the content items O_o, or the rows indexes of the **PIT**, for which there are **no** pending Interests, in 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. We diagonalize the result, so that it is appropriate for multiplication with $\mathbf{A}'$. In line 5, we keep the rows of $\mathbf{A}'$ with such indexes, 'erasing' the remaining rows (i.e. setting their elements to 0). These are saved in $\mathbf{B}$, which consist in the only Interest signals that the NDN router needs to forward upstream. Note that even if $\mathbf{A}'$ 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 $\mathbf{B}$, resulting from the use of binary operations, neatly serves our purposes. In line 6, the contents of the **PIT** are updated by performing a logic OR, '|', with $\mathbf{A}'$, 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.

[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 3. This dual representation is extended to the FIB and CS.

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

1: **define** updateOnData(**A**):
2:
3: $\mathbf{A}' \leftarrow \mathbf{A}(|O|+1:2\,|O|,:)$
4: $\mathbf{D} \leftarrow \mathbf{A}' \times \mathbf{1}$
5: $\mathbf{E} \leftarrow \text{diag}(\mathbf{D}) \times$ **PIT**
6: **PIT** $\leftarrow \text{diag}(\neg\mathbf{D}) \times$ **PIT**
7: **return E**

With the explanation given above for the **updateOnInterest()** routine, the understanding of **updateOnData()** is left to the reader. Note that the objective of this last routine is the reverse operation of **updateOnInterest()**, i.e. forwarding Data packets over the interfaces for which there is a registered Interest in the **PIT**. We also note that the final step of the operation (line 6) consists in erasing all Interest registrations which have been successfully attended.

2) 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

$$\mathbf{FIB} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{5}$$

The **FIB** encoding shown in 5 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 the Interest signals, **B**, returned by the **updateOnInterest()** operation:

1: **define** forwardInterests(**B**):
2:
3: $\mathbf{B}' \leftarrow \mathbf{B} \times \mathbf{1}$
4: $\mathbf{L} \leftarrow \text{diag}(\mathbf{B}) \times$ **FIB**
5: $F_{n,1:|F|,out} \leftarrow \mathbf{L}$

The last operation of the **forwardInterests()** routine (line 5) refers to the (parallelized) 'filling' of the output ports of all the interfaces F of router R_r.

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

3) 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 6 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].

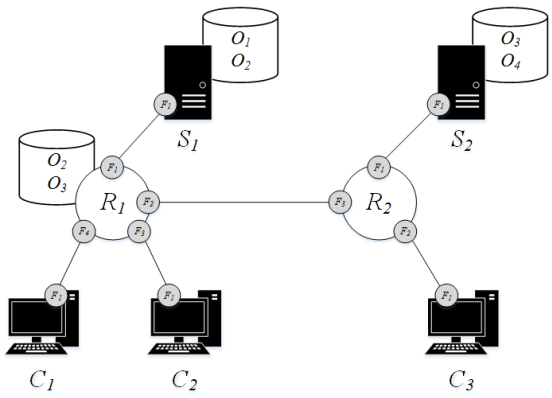

Fig. 3: Simple example of an NDN network topology.

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

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 '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 caching algorithm implemented by the CS. In fact, we model a CS as having a general behavior (presented in this section), and a specific behavior, which depends on the cache algorithm the CS implements. We discuss caching algorithms in Section III-D.

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:

$$\sum_{o=1}^{|O|} \mathbf{CS}_{o,p} \leq 1 \quad \forall\ p \in 1, 2, ..., |P| \tag{7}$$

$$\sum_{p=1}^{|P|} \mathbf{CS}_{o,p} \leq 1 \quad \forall\ o \in 1, 2, ..., |O| \tag{8}$$

C. Endpoints

D. Caching Algorithms

E. 'Connecting the Dots'

We represent a network topology using a square matrix **T**, with dimension $|R| + |C| + |S|$. For representation purposes,

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

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 **T** shown in 9 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.

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 4 & 3 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{9}$$

IV. Implementation

This section provides noteworthy details on our implementation of the methods introduced in Section III.

A. Heading 1

V. Experiments

A. Experimental Setup

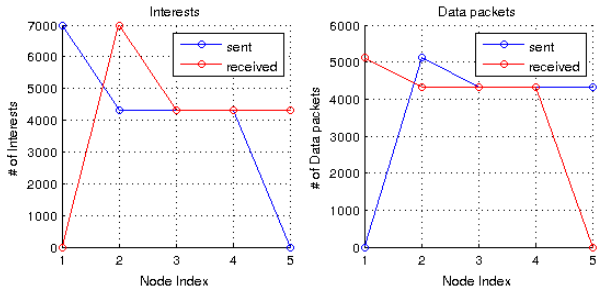

Fig. 4: Graphical depiction of our NDN router model.

VI. 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.

[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.