sensors

MDPI

Article

Interest Forwarding in Named Data Networking Using Reinforcement Learning

Olumide Akinwande

Department of Electrical and Electronic Engineering, Imperial College, London SW7 2AZ, UK; oja13@imperial.ac.uk; Tel.: +44-7860847129

Received: 14 August 2018; Accepted: 28 September 2018; Published: 8 October 2018

Abstract: In-network caching is one of the key features of information-centric networks (ICN), where forwarding entities in a network are equipped with memory with which they can temporarily store contents and satisfy en route requests. Exploiting in-network caching, therefore, presents the challenge of efficiently coordinating the forwarding of requests with the volatile cache states at the routers. In this paper, we address information-centric networks and consider in-network caching specifically for Named Data Networking (NDN) architectures. Our proposal departs from the forwarding algorithms which primarily use links that have been selected by the routing protocol for probing and forwarding. We propose a novel adaptive forwarding strategy using reinforcement learning with the random neural network (NDNFS-RLRNN), which leverages the routing information and actively seeks new delivery paths in a controlled way. Our simulations show that NDNFS-RLRNN achieves better delivery performance than a strategy that uses fixed paths from the routing layer and a more efficient performance than a strategy that retrieves contents from the nearest caches by flooding requests.

Keywords: information centric networks; named data networking; cognitive packet networks; random neural networks

1. Introduction

Information-centric networking (ICN) has become one of the leading topics in the next-generation networking and future Internet architecture research. The fundamental concept in ICN is to make content directly addressable, a significant departure from the host-centric communication model of the current Internet. To achieve this, ICN names and addresses data units at the network level, so that in the place of a host address, request packets are routed and satisfied based on the name or identification of the desired content.

This approach is motivated by the increasing domination of Internet traffic by multimedia, effectively making the Internet a content distribution network [1]. Despite the success of the current solutions for efficient information delivery like the content delivery networks (CDNs) and peer-to-peer (P2P) networking, they are limited and complicated by the underlying host-to-host communication [2]. By treating data as fundamental to the architecture, ICN seeks to match the current demands being placed on the Internet and improve on these patch solutions. The separation of content from location enables ICN to support naturally desirable features like in-network caching and multicast communication. In addition to the efficient delivery of content, ICN, using data-centric methods, also addresses significant issues including mobility, security, and quality of service.

Various projects explore the ICN concept, differing mainly in their formats for naming data, caching policies, routing of data objects and requests, and in how they handle mobility and security [3,4]. Of particular interest to us in this work, is the Named Data Networking architecture [5,6] which incorporates a stateful and intelligent forwarding plane in addition to a routing layer.

The forwarding plane, also called the *strategy module*, is expected to leverage on both the information provided by the routing layer and the observed packet delivery measurements to make adaptive decisions when forwarding requests. In the general case, the strategy module is defined for each reachable name announced by the routing layer. Furthermore, such an intelligent forwarding plane could relax the stringent convergence and correctness demands commonly placed on the routing layer as explored by the work in [7].

For scalability and practicality reasons, the name-based routing protocol in NDN can only announce and monitor paths leading to the actual publishers and designated repositories, thus leaving the responsibility of exploiting the in-network caching capability, which is vital for delivering the design goals of NDN, entirely to the forwarding strategy. However, most of the proposals for the strategy layer, including the default algorithm adopted by the NDN project, follow a "monitor-the-routing-paths" approach, whereby their forwarding and probing actions are restricted to the interfaces suggested by the routing layer cite [7–11]. This approach limits the forwarding strategy from exploiting local caches closer but outside the routing paths.

In this paper, we develop a dynamic self-aware [12] strategy layer for NDN architectures to offer fast content delivery using local content stores, and we also keep the existing capabilities of the routing layer. The NDN forwarding strategy that we implement exploits the Random Neural Network (RNN) [13] with Reinforcement Learning, similar to the Cognitive Packet Network (CPN) [14,15]. An overlay network with a similar scheme is described in [16].

The rest of the paper is organised as follows. In the remainder of this section we briefly describe the NDN forwarding engine. In Section 2, we review related work on forwarding in NDN, while Section 3 presents a reinforcement learning algorithm for the RNN [13,17]. The *NDN forwarding strategy using reinforcement learning with the RNN* (NDNFS-RLRNN) is detailed in Section 4. In Section 5, we describe the system we evaluate in our simulations, and we discuss our results in Section 6. Finally, conclusions and future work are discussed in Section 7.

The NDN Forwarding Plane

NDN uses two types of packets for communication. Requests are issued using *Interest packets*, while *Data packets* are used to carry the requested contents. Both packets include the name or identification of the desired data object. An Interest is uniquely identified by its name and nonce values, where the nonce is randomly generated by the requesting application. In addition, the Data packet also includes a digital signature, created by the content producer, that binds the name with the content.

An NDN content router (CR) maintains three data structures for implementing the forwarding plane: a *content store* (CS), a *pending interest table* (PIT) and a *forwarding information base* (FIB). The CS acts as a temporary cache for Data. It is used to implement in-network caching and can satisfy content requests. A CR caches traversing Data packets according to a well-defined caching policy. The PIT is used to keep track of the unanswered Interests. The PIT also aggregates information about subsequent Interests that arrive and match any of its entries; the CR then decides whether or not to forward such Interests. The information recorded in a PIT entry will include the content name, a list of arrival and outgoing links, and the arrival and forwarding times of the Interests. Since Interests can be uniquely identified, the PIT can detect when an Interest loops. Moreover, by keeping the forwarding state in the PIT, delivery performance can be estimated and used by the strategy module in influencing future forwarding decisions. Finally, the FIB contains a mapping of the reachable content names to the outgoing interfaces of the CR. The FIB is populated and updated by a name-based routing protocol. The strategy module uses the routing information in the FIB in forwarding Interests.

It is important to mention that the format for naming data objects is an active area of research in ICN [3,18]. NDN adopts a hierarchical structure in naming content which is similar to URLs. When a CR receives an Interest, it first checks its CS for matching data objects. According to the blueprint for the NDN architecture in [8], Data in the CS is said to match an Interest if the content name

of the Interest is a prefix of the content name of the Data. NDN also allows the requesting application to have some control over which Data packets are satisfactory through an optional *selector* field in the Interest packet. In the event of a match, the Data packet is returned on the arrival interface of the Interest packet. Otherwise, a lookup, which should also consider the entries in the selector field, is performed in the PIT for an exact match. A match in the PIT means the CR is already expecting a response that will satisfy the received Interest, so the corresponding PIT entry is updated and the strategy module decides whether or not to forward the Interest packet. Otherwise, the longest-prefix match entry in the FIB is found, and the strategy module decides the outgoing interface(s) to forward the Interest. If there is no match in FIB, the CR has no knowledge of the desired content, so the Interest is discarded.

Data packets, on the other hand, are forwarded along the reverse path used by their corresponding Interest packets by using the information recorded in the PITs. On receiving a Data packet, the CR performs a lookup in the PIT for a matching entry. If there is a match, the CR sends a copy of the Data packet on all the interfaces listed under "incoming" in the entry. The CR then deletes this entry from the PIT and decides whether or not to cache the Data packet. Otherwise, the content is considered unsolicited, and the Data packet could be discarded without caching.

2. Related Work

It is shown in [19] that coupling caching and forwarding is essential in ICN to significantly benefit from ubiquitous caching. The authors first studied an optimal policy, the *ideal Nearest Replica Routing* (iNRR), in which CRs forward their requests to the nearest possible replica with the help of an "oracle" that keeps track of the network's caching state. Then, due to the cost of realising such an oracle, they proposed practical implementations where CRs periodically explore a given neighborhood in the network by flooding the request. A review of caching strategies that have been evaluated in the ICN research is presented in [20].

In the NDN context, Jacobson et al. [8] proposed forwarding an Interest along all the interfaces suggested by the routing layer excluding the interface the Interest arrives on, referred to as the *multicast strategy*. The *best route strategy* [9] forwards interests using the available upstream with the lowest routing costs. The Adaptive SRTT-based Forwarding Strategy (ASF) is proposed in [7]. ASF sends an Interest using the upstream with the lowest measured SRTT among those put forward by the routing protocol. To gather measurements, ASF also probes alternative interfaces at intervals. The current NDN forwarding strategy [6,10,21] combines interface ranking and colour classification to decide where to forward Interests. Interfaces suggested by the routing protocol are first classified using a colour scheme according to how well they are known to return Data, then the interfaces are ranked in each class using some metric, usually the smoothed round trip time (SRTT). The forwarding logic is to use the highest ranked available interface in the best possible classification. NDN also introduced Interest to address the inefficiencies that result from the dangling states in the PIT caused by unsatisfied Interests. When it cannot forward nor satisfy an Interest, a CR responds with an Interest NACK; the Interest NACK also carries a code describing the reasons. Therefore, Interest NACKs can help the network to quickly and informedly detect faults and, when possible, try other forwarding options.

To eliminate undetected loops in NDN, the *Strategy for Interest Forwarding and Aggregation with Hop-Counts* (SIFAH) is proposed in [11]. In SIFAH, a CR accepts to forward an Interest only if there exists, based on distance information to the content source, a neighbour node that moves the packet closer to the content.Otherwise, an Interest NACK is returned. The distance information used in SIFAH is the number of hops to the content repository, and it is provided to the forwarding plane by the routing protocol. Also, by replacing nonce values with distance information, SIFAH reduces the memory overhead incurred by the PIT. While SIFAH guarantees a correct forwarding strategy, it achieves this by limiting the dynamism of the forwarding plane in making adaptive decisions. In addition, the conditions it uses for loop detection, and therefore for forwarding Interests, are sufficient conditions.

This means that an interface can fail these conditions and yet not lead to an Interest loop being formed. Hence, the possibility of unnecessarily denying service to Interests exists.

Another class of forwarding strategies referred to as the *multipath* strategies [22–24], dynamically assign a forwarding weight or probability to each interface of a CR, which determines the proportion of Interest traffic sent on it. The main aim here is to achieve load balancing and manage congestion in the network.

In our work, we propose implementing the strategy module of the NDN architecture using an online learning algorithm. Reinforcement learning has been previously proposed for the NDN forwarding strategy. In [25] the *multi-armed bandits strategy* (MABS) is developed which assumes no knowledge of path information from the routing layer. As a result, when no forwarding information is available, the CR floods a request on all its interfaces. *INFORM*, presented in [26] and inspired by the Q-routing algorithm [27], adopts a more similar approach to ours by leveraging on the routing layer. INFORM alternates between exploration and exploitation phases when making forwarding decisions. In the initial exploration phase when no learning has occurred, a received Interest is sent using the best interface according to the information in the FIB, and a copy of the Interest is also sent on a randomly selected interface. The same actions are repeated in subsequent exploration phases except that the best interface will be the one learnt by the algorithm. Only the best interface computed after an exploration phase is used during the subsequent exploitation phase. In these methods, the authors do not address the handling of Interest NACKs, which becomes significant because the exploration of the network, necessary for the learning algorithm, increases the possibilities of Interest loops.

Our approach is inspired from the CPN routing protocol which, in most of its implementations, employs a reinforcement learning algorithm using the RNN [13] to establish and maintain connections between network nodes. The algorithm used has been shown to possess fast convergence properties during an initial learning phase and good sensitivity to environmental changes [17], hence its success in CPN routing. An early review of the variations, applications, and performance evaluations of the CPN can be found in [28].

3. Reinforcement Learning and the Random Neural Network

The RNN is a neural network model inspired by queueing networks [29] where neurons send and receive both positive and negative signals. The positive signals are called *excitatory*, while the negative signals are called *inhibitory*. It has been used in many applications, including image and video compression [30,31], the recognition of tumours from Magnetic Resonance Images of the human brain [32], toxicity prediction of chemical compounds [33], web search [34], and network routing and cloud management [35–39]. It is a special instance of the family of stochastic networks known as G-Networks [40–45] which have many different areas of application.

Table 1. Notation for the RNN model.

Notation	Definition
$k(t)$	State vector of the RNN at time t where $k_i(t) \geq 0$ is the potential level of a neuron i in the network
r_i	Firing rate at neuron i
Λ_i	Arrival rate of positive exogenous signals at neuron i
λ_i	Arrival rate of negative exogenous signals at neuron i
$p^+(i,j)$	Probability that a signal leaving neuron i is excitatory and heads for neuron j
$p^-(i,j)$	Probability that a signal leaving neuron i is inhibitory and heads for neuron j
$d(i)$	Probability that a signal leaving neuron i departs the network

Table 1 lists the notation for the RNN model. The state of a neuron i in the network at any time is represented by a non-negative potential value $k_i(t)$, so that the vector:

$$k(t) = (k_1(t), k_2(t), ..., k_n(t))$$

describes the state of an RNN at any time t. An incoming signal either increases a neuron's potential by one if it is excitatory or decreases it by the same amount if it is inhibitory. An excited neuron i, that is, one with non-zero potential ($k_i(t) > 0$), fires randomly by sending out excitatory or inhibitory signals to other neurons or to the outside of the network. Whenever a neuron fires, its potential reduces by one. Signals can also arrive at a neuron from outside the RNN.

For each neuron i in an n-neuron network:

$$\sum_{j}^{n}[p^+(i,j) + p^-(i,j)] + d(i) = 1. \tag{1}$$

It was shown in [46] that the RNN with exponential firing intervals and Poisson exogenous signal arrivals has the product form solution:

$$p(k) = \prod_{i=1}^{n}(1-q_i)q_i^{k_i},\ q_i = \frac{\lambda_i^+}{(r_i + \lambda_i^-)}, \tag{2}$$

where $p(k) = \lim_{t\to\infty} Pr[k(t) = k]$, $k = (k_1, k_2, ..., k_n)$, is the network's stationary probability distribution; q_i is the steady state probability that neuron i is excited. λ_i^+ and λ_i^- represent the overall flow of positive and negative signals, respectively, at neuron i. For $i = 1, 2, ..., n$, λ_i^+ and λ_i^- satisfy the following system of non-linear equations whose solution exists and is unique [47]:

$$\lambda_i^+ = \sum_j [q_j r_j p^+(j,i)] + \Lambda_i \tag{3}$$

$$\lambda_i^- = \sum_j [q_j r_j p^-(j,i)] + \lambda_i \tag{4}$$

For adaptive routing applications, the recurrent RNN model is normally used. This RNN is fully connected and has as many neurons as there are possible outgoing links at the node it resides, that is, a neuron represents a possible forwarding decision. The weight of each RNN connection (i,j) is the emission rate of positive or negative signals from neuron i to neuron j. That is,

$$w^+(i,j) = r_i p^+(i,j), \quad w^-(i,j) = r_i p^-(i,j)$$

Introducing the above expressions and assuming signals do not leave the network ($d(i) = 0$), we can rewrite Equation (1) as:

$$r_i = \sum_{j=1}^{n}[w^+(i,j) + w^-(i,j)] \quad i = 1, 2, ..., n \tag{5}$$

The matrices of the weights of the connections: $W^+ = \{w^+(i,j)\}$, $W^- = \{w^-(i,j)\}$, are important parameters whose values are continuously modified during the learning process.

The RNN is initialised with all the weights equal. This is done by choosing a firing rate r_i and using Equation (5) to solve for the initial weight value. That is, $w = \frac{r_i}{2(n-1)}$, where w is the initial value of all weights; remember that $w^+(i,i) = w^-(i,i) = 0\ \forall i$.

The pseudocode in Algorithm 1 illustrates how reinforcement learning can be implemented using the RNN. The reinforcement learning algorithm used is inspired from the *E-rule* scheme in [17], and it assumes that a reward value can be estimated for every decision. In this approach, the outcome of a previous decision, R_l, is compared with an internal expectation, T_{l-1}, of the neural network and the RNN receives reinforcement based on the result of this comparison. The reinforcement can either be *positive* if the outcome is considered a success, or *negative* otherwise. In the illustration in Algorithm 1, the l-th decision is considered to be a success if $R_l \geq T_{l-1}$; otherwise it is a failure. A success leads to a significant increase in the excitatory weights going into the corresponding neuron and a small

increase in the inhibitory weights leading to the other neurons. Otherwise, the inhibitory weights of the neuron assigned to the decision is significantly increased and the excitatory weights into the remaining neurons are slightly increased.

Algorithm 1 The RNN training process

INPUT: T_{l-1}, W^+, W^-, internal expectation and weight matrices before the l-th reward is received; $0 < \alpha < 1$.

INPUT: $R^i{}_l$, the l-th received reward obtained when the lth decision was to select output link i.

OUTPUT: $q_i, \forall\, i = 1, 2, ..., n$, the excitation probabilities for an RNN with n neurons

if $R^i{}_l \geq T_{l-1}$ **then**
  /* l-th decision is a success*/
  for each neuron j in the RNN, $j \neq i$ **do**
    $w^+(j,i) \leftarrow w^+(j,i) + R^i{}_l$
    for each neuron k in the RNN, $k \neq j$ **do**
      $w^-(k,j) \leftarrow w^-(k,j) + \frac{R^i{}_l}{n-2}$
else
  /* l-th decision is not a success*/
  for each neuron j in the RNN, $j \neq i$ **do**
    $w^-(j,i) \leftarrow w^-(j,i) + R^i{}_l$
    for each neuron k in the RNN, $k \neq j$ **do**
      $w^+(k,j) \leftarrow w^+(k,j) + \frac{R^i{}_l}{n-2}$
/* Normalizing the weights */
for each neuron j in the RNN **do**
  $r_j^* \leftarrow \sum_m^n [w^+(j,m) + w^-(j,m)]$
  for each neuron k in the RNN, $k \neq j$ **do**
    $w^+(j,k) \leftarrow w^+(j,k) * \frac{r_j}{r_j^*}$
    $w^-(j,k) \leftarrow w^-(j,k) * \frac{r_j}{r_j^*}$
/*solve the n non-linear simultaneous equations in (2) for each $0 \leq q_j \leq 1$ */
/* We use a fixed-point iteration, starting with $q_j^0 = 0.5\ \forall\, j = 1, 2, ..., n$ */
$q_j^{k+1} \leftarrow min[1, \frac{\sum_m [q_m^k w^+(m,j)] + \Lambda_j}{r_j + \sum_m [q_m^k w^-(m,j)] + \lambda_j}]$
/* Updating the internal expectation */
$T_l \leftarrow \alpha T_{l-1} + (1-\alpha) R_l$ /* α is to be chosen closer to 1 */

After updating the weights, Algorithm 1 also shows a renormalisation of the weights. This is to prevent ever-increasing weights, and the fact that it is the relative magnitudes of the weights that determine the state of the network justifies this step. The new firing rate r_i^* for each neuron is first evaluated using the updated weights in Equation (5). Then, for each neuron i the normalised weights are computed as $w^+(i,j) = w^+(i,j) * \frac{r_i}{r_i^*}, w^-(i,j) = w^-(i,j) * \frac{r_i}{r_i^*},\ j = 1, 2, ..., n$, where r_i is the initial or reference firing rate.

Finally, the excitation probabilities q_i for all the neurons are evaluated by solving the non-linear system of equations $q_i = \frac{\lambda_i^+}{(r_i + \lambda_i^-)}, \lambda_i^+ = \sum_j [q_j r_j p^+(j,i)] + \Lambda_i, \lambda_i^- = \sum_j [q_j r_j p^-(j,i)] + \lambda_i$.

The common approach to solving these equations is using the simple fixed-point iteration as shown in Algorithm 1. At any given time, the neuron with the highest excitation probability q is considered the best decision.

Finally, the internal expection is updated with the new reward using $T_l = \alpha T_{l-1} + (1-\alpha)R_l$, where $0 < \alpha < 1$.

4. NDN Forwarding with Reinforcement Learning Using the RNN

We present NDNFS-RLRNN that uses reinforcement learning (RL) with the RNN for the strategy module per prefix in the NDN architecture, which operates with a form of smart Opportunistic Communication [48], supported by a name-based routing protocol with online measurements from the state recorded in the PIT. Our goal is to exploit the convergence properties of the RL with RNN algorithm which have been well studied in [17,49] algorithm to effectively search for local content store hits.

In NDNFS-RLRNN, an RNN is created for a prefix in the FIB and its creation is triggered by a new Interest for the corresponding prefix. In its initial state, the RNN only knows of the routing preferences for its prefix and is yet to be updated by packet delivery measurements. Here, NDNFS-RLRNN's forwarding decision will be to use the best interface according to the routing layer. Each RNN per prefix at the CR has as many neurons as the number k of outgoing links of the router, and the RNN has k^2 weights, and if the router has n_D active destinations, the router will have a total of n_D RNNs to handle each of the destinations, or a total of $n_D \times k^2$ entries. A conventional routing table at the NDN router will have $n_S \times n_D \times k$ entries for n_S active sources. Thus when $n_S > k$, our scheme uses a smaller data structure per router than a conventional NDN scheme [50].

On the arrival of a Data packet, a reward value is computed for the arrival interface and used to update the RNN as illustrated in Algorithm 1. The goal of our distributed reinforcement learning is to minimise the delay for retrieving contents, so we estimate reward using the RTT values. Let ${T^j}_I$ be the time an NDN router forwards an Interest along an interface j and ${T^j}_D$ be the arrival time of the Data packet satisfying the Interest which also arrives on the same interface, we can estimate the reward, R^j as the inverse of the RTT using

$$R^j = ({T^j}_D - {T^j}_I)^{-1} \tag{6}$$

For an updated RNN, the forwarding decision of NDNFS-RLRNN is, excluding the arrival interface, with a high probability p only the interface corresponding to the most excited neuron is used or with probability $(1-p)$ a probing decision is made. We refer to p as the *probe parameter*. Before explaining the probing process, we first introduce the idea of marking an Interest packet for probing. This means an Interest can be identified at the CRs as either a "normal" Interest or a probe Interest. The probing process involves sending two packets: the original Interest which is sent along the best known interface and a copy which is sent as a probe Interest on a randomly selected interface. We impose that a CR only forwards a probe Interest along the best known interface. The motivation for this probing process is to control exploration in order to reduce the possibility of unnecessarily using longer paths to retrieve contents and to manage overhead. In Figure 1 we use a simple flowchart to summarise the Interest forwarding and probing process of NDNFS-RLRNN.

We have also adopted the use of Interest NACKS as in the current NDN in our approach. When a CR can neither satisfy nor forward an Interest, it responds using an Interest NACK. Clearly, the reward estimation is explicit when an Interest is satisfied. However, this is not the case when an Interest NACK is returned or when no response is received before the Interest times out. In such cases, the reward can be estimated as:

$$R^j = [c(T - {T^j}_I)]^{-1}, \tag{7}$$

where T is the current time and c is a constant large enough such that a negative reinforcement is applied. Therefore, before deleting a PIT entry, all the interfaces that failed to return data will receive negative reinforcement based on the reward computed in Equation (7). On the arrival of an Interest NACK in response to a request, the CR will first try to resend the Interest using the routing layer preferences. If retransmission is impossible, an Interest NACK is returned on all the expecting interfaces, and the RNN for this prefix returns to the initial state.

To reduce the inefficiencies that could be introduced by out-of-date information and for efficient use of resources, when an RNN receives no feedback for T_r time units, it is deleted. As a result, an RNN will remain in the system only if it is considered active according to the above T_r condition. This is necessary to manage the limited resources at each CR. Furthermore, a limit could be set for the number of RNNs at a node depending on the available network resources, such that new Interests that arrive after this limit is reached will be sent using the routing layer options without any attempt for a local search.

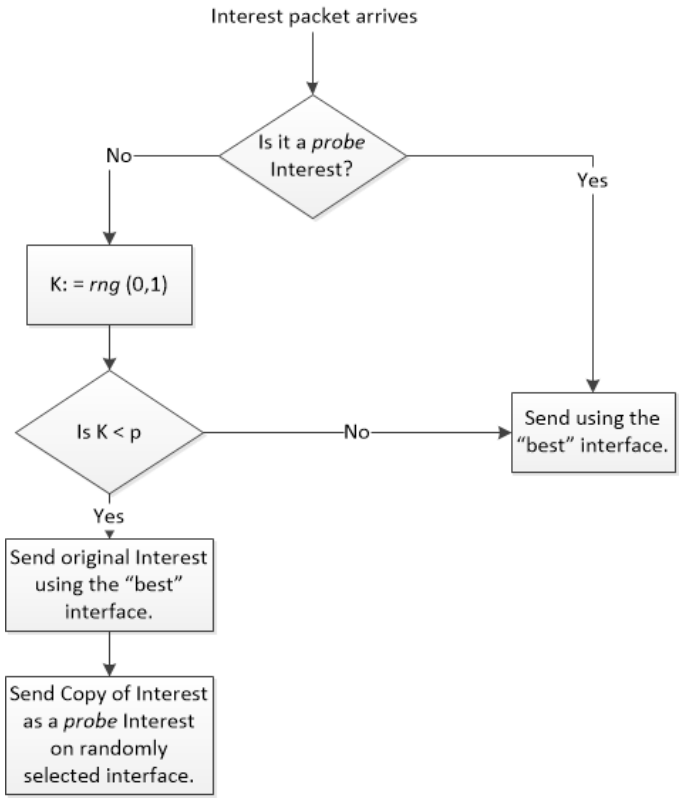


Figure 1. Interest forwarding and probing process of NDNFS-RLRNN. It uses the ϵ-greedy approach for the probing. *rng*(0,1) is a random number generator that returns a number in the range $[0, 1)$. p is the probe parameter of the RNN. It is assumed that the RNN is no longer in its initial state and that forwarding is possible so that the best interface refers to the interface corresponding to the most excited neuron of the RNN.

5. System and Scenario Description

While there is no complete consensus on the evaluation of information-centric networks, the scenarios we consider in our simulations are in line with much of the related literature, especially with respect to the workload and the network topology used. In the remainder of this section, we describe the models and the assumptions adopted for the system we analyze.

5.1. Request Arrival Model

The most common approach in analysing caching system is the assumption of the independent reference model (IRM) [51] to model the arrival of requests, with a Zipf-like distribution to represent the content popularity. IRM describes the requests arriving at a cache for a fixed catalog of items as forming a sequence of independent, identically distributed random variables. In other words, the probability that a request is for an item is a constant and independent of the past requests.

The IRM has the advantage that it is simple and has enabled the development of tractable models in the analysis of caching systems [52–54]. Although IRM does not adequately capture

phenomena like *temporal locality* and *spatial locality* which have been shown to exist in real traces [55,56], other studies [57,58] argue that their long-term effect can be well accounted for by careful choice of a Zipf-like distribution to represent content popularity. Temporal locality occurs when recently accessed items are more likely to be requested in the near future while spacial locality describes the situation where the request for an item becomes more likely because a “similar" item was referenced in the recent past.

5.2. Content Popularity

The choice of the popularity distribution is an important factor that determines the performance of in-network caching in ICN. The Zipf's law is the favoured choice to model popularity in ICN [19,59,60].

Zipf's law, popularized initially in Statistical Linguistics, predicts, for a catalog of N items, that the probability of referencing the i-th most popular item is:

$$\rho_N(i) = \frac{\frac{1}{i^{\alpha}}}{\sum_{i=1}^{N} \frac{1}{i^{\alpha}}}, \tag{8}$$

where the *skew factor*, α, is a constant characterising the distribution.

The more general Mandelbrot-Zipf (Mzipf) distribution is also adopted in some analysis [61,62]. In MZipf,

$$\rho_N(i) = \frac{\frac{1}{(i+q)^{\alpha}}}{\sum_{i=1}^{N} \frac{1}{(i+q)^{\alpha}}}, \tag{9}$$

where α and q both characterise the distribution. Equation (9) reduces to (8) when $q = 0$.

There is also no consensus on the parameter settings for either distributions. However, the values used in the literature are informed by those arrived at from large-scale studies of actual traces observed at ISPs and CDNs [57,63–65]. As a result, values of $\alpha \in [0.6, 2.5]$ and $q \in [0, 50]$ are common, while [24,66,67] include uniform popularity ($\alpha = 0$) in their evaluations.

In our tests, we adopt the MZipf distribution to represent content popularity, and we use values within these ranges which are realistic, considering the catalogue sizes we consider, to characterise the distribution.

5.3. Network Topology

With respect to the network segment used in our tests, we have also followed the trend in recent ICN studies by adopting a real ISP topology. Furthermore, we assume that the nodes and links within the topology have identical resources in terms of cache allocation and link capacity. Finally, we assume in all the tests carried out that the system operates below congestion.

6. Results and Discussion

In this section, we present initial evaluations of the performance of NDNFS-RLRNN through extensive simulations using the *ndnSim* [68], an NS-3-based simulator which already exists for NDN.

For the simulations, we consider an NDN-based network connecting users to content servers. We use the real topology *Elibackbone* shown in Figure 2 according to the dataset in [69]. We consider a single access router for the network through which requests are served from the content servers. At each node, except the access node, external request arrivals are Poisson process with a rate 5 request packets per second, and each data object is 10 KB in size. All the simulations begin with empty caches at the nodes.

Furthermore, we compare NDNFS-RLRNN with the *Adaptive SRTT-based Forwarding Strategy* (ASF) [7], and a Nearest Replica Routing (NRR) strategy [19]. The *ndnSim* has the ASF algorithm pre-installed. The ASF strategy uses the upstream with the lowest measured SRTT and probes alternative interfaces suggested in the FIB at intervals. The length of the probing interval is reduced

from the default value of 60 s to 3 s to increase probing, and we install all possible routes in the FIB such that no loop exists in the forwarding. For the NRR strategy, we implement a multicast algorithm that sends each request on all the interfaces of a node except the arrival interface, which guarantees that the requested data objects are retrieved from the closest caches. Other approaches are possible where the flooding is periodic as suggested in [19], but we have chosen this simple method that avoids parameter settings. The forwarding strategies are implemented per content in the catalogue.

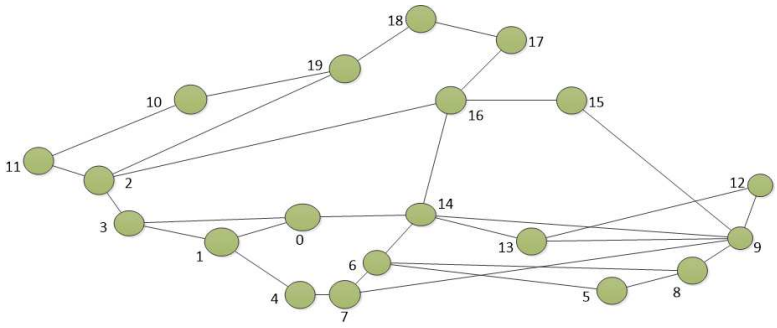


Figure 2. *Elibackbone* topology. The network consists of 20 nodes and 30 links.

Since the cache states in the network will influence the performance, the forwarding strategies are compared under three different caching policies from the literature:

- Leave copy everywhere combined with LRU replacement policy (*LCE*): Here, the CRs cache every data packet received.
- Betweenness centrality policy and LRU (*Betw*): In *Betw*, proposed in [70], only the routers with the highest betweenness centrality values along the delivery path cache the content. These values are computed offline using the betweenness centrality definition in graph theory.
- *ProbCache and LRU*: Here the decision to cache a content at a CR is based on a defined probability. We set this probability using the proposed *ProbCache* Algorithm [71], which computes the caching probability when a content arrives at a node by weighing the cache capacity along the remainder of the delivery path with the relative position of the router along this path, as shown in Equation (10).

$$p(x) = \frac{\sum_{i=1}^{c-x+1} N_i}{kN} \tag{10}$$

where c is the total number of content routers on the path from requester to content source, x is the position of the current node along this path, N_i is the cache capacity at node i, N is the average cache capacity along the full path, and k is a constant denoting a target cache capacity along a path.

LCE and *ProbCache* are ubiquitous caching policies differentiated by their cache eviction rates. By ubiquitous, we mean that nodes in the network are not prioritised in any way when caching. *Betw*, on the other hand, tries to cache predominantly at the most “central" nodes in the network. These policies represent three categories of on-path caching strategies in the literature.

To evaluate the performance of the forwarding strategies, we measure and report the *cache hit rate* and the *network load per request*. The cache hit rate is measured as the proportion of the requests arriving into the network which are satisfied from the local caches. On the other hand, the network load or overhead is the total number of hops traversed by the network packets, which includes Interests, Data packets and Interest NACKs, per request sent into the network.

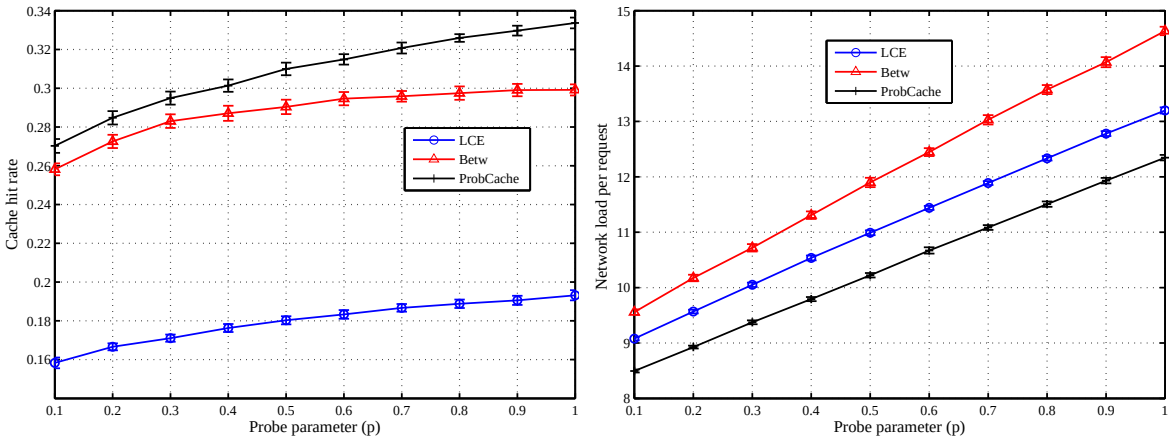
Finally, we run each simulation scenario for 400 s, and all metrics are obtained through the mean and standard deviation of 20 randomised simulation runs.

6.1. Probe Parameter

We investigate the impact of the probe parameter (p) of the NDNFS-RLRNN algorithm. This parameter determines how often alternative paths are explored by a CR. In this simulations, we consider a uniform content popularity model across the network with Mzipf parameters set at $\alpha = 1.0$ and $q = 5.0$. By uniform popularity, we mean that request arrivals at each CR in the network are characterized by the same MZipf parameters. We fix the catalogue size at 1000 data objects, and each CR is equipped with a cache size of 1% of the catalogue.

In Figure 3, we observe that for uniform content popularity in the network, increasing the exploration of the network, as a result of increasing p, increases the cache hit rates at the cost of increased network overload. In general, the *ProbCache* policy produces the best cache hit performance because it combines better redundancy and eviction properties compared with the other policies. The LCE produces the worst performance because of its characteristic high redundancy and high eviction rate. Although *Betw* reduces redundancy compared with LCE, there is a high rate of eviction at the most "central" nodes.

Interestingly, while we observe a steadier increase in the hit rates as p increases for both the LCE and *ProbCache* policies, for *Betw*, the benefit reduces as p is increased. In fact, 90% of the improvement from increasing p from 0.1 to 1.0 is already attained at $p = 0.6$ for Betw compared with about 70% for both LCE and *ProbCache*. Since *Betw* caches contents mostly at specific nodes, NDNFS-RLRNN can search these locations considerably well enough such that further increasing exploration becomes less cost-effective. As expected, Figure 3b shows the overhead cost of increasing the probe parameter.



(**a**) (**b**)

Figure 3. Evaluation of the probe parameter (p) under different on-path caching policies. (**a**) Cache hit rate; (**b**) Network load per request. Each CR is equipped with a cach size of 1% of the content catalogue. Content popularity is modelled according to the Mandelbrot-Zipf (MZipf) distribution with the settings: $\alpha = 1.0$ and $p = 5.0$. The results reported are the average of 20 randomized simulation runs.

6.2. Catalogue Size

In Figures 4 and 5, we compare the three forwarding strategies and show the impact of the cache size to the catalogue size ratio. In general, performance diminishes as this ratio reduces. We observe that with low exploration, NDNFS-RLRNN can exploit in-network caching better than ASF for all the considered scenarios. With respect to the cache hit rate, it provides a performance improvement compared with ASF of between 23–30%, 20–32%, and 22–30% under the LCE policy, *Betw* policy, and *ProbCache* policy, respectively. Also, NDNFS-RLRNN provides an improvement under the *Betw* policy compared with the NRR strategy of between 3–16%. It achieves these improvements with less overhead as shown in Figure 5c; and as expected, the overhead NRR incurs is significantly higher than the other two forwarding strategies.

The network overhead result as shown in Figure 5 is a measure of the cost to network incurred by the forwarding strategies. The probing of interfaces introduces additional costs because it increases the number of Interests processed in the network and, as a result, the number of Data packets and Interest NACKs. As a result, NRR incurs significantly higher overhead compared with the other forwarding strategies because it tries to flood the network with every request. The ASF is similar, in terms of probing, to the NRR except that the probing is periodical and restricted to a subset of the interfaces at each CR. This explains why NRR generates more than 2.5 times the overhead for ASF in all the considered scenarios. Moreover, NDNFS-RLRNN incurs the least overhead among the considered forwarding strategies primarily because of how it tries to take advantage of the probing process. Also, the low probing parameter value of 0.3 and the use of *probe* Interests to control the network exploration are contributing factors.

Finally, the relatively high deviation from the mean values, compared to the other results, can be explained by the randomization of the content distribution at the CRs.

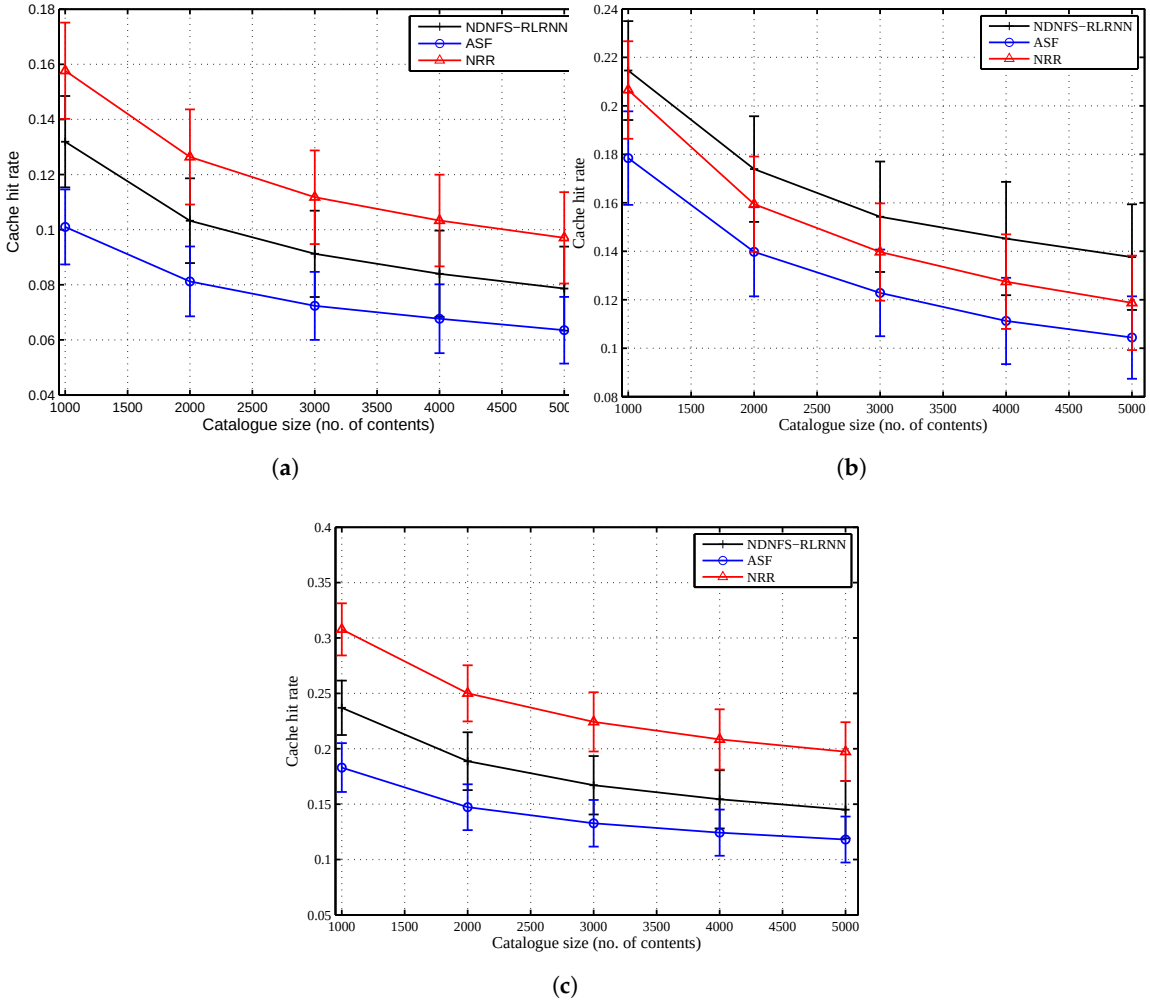


Figure 4. Cache hit rate performance of NDNFS-RLRNN ($p = 0.3$), ASF and NRR for different cache size to catalogue size ratios, under: (**a**) LCE caching policy, (**b**) *Betw* caching policy and (**c**) *ProbCache* caching policy. The cache size of each CR is fixed at 10 data objects while we vary the catalogue size fom 1000 to 5000 data objects. Content popularity is modelled according to the Mandelbrot-Zipf (MZipf) distribution with the settings: $p = 5.0, \alpha \in [0.6, 1.2]$. The results reported are the average of 20 randomized simulation runs.

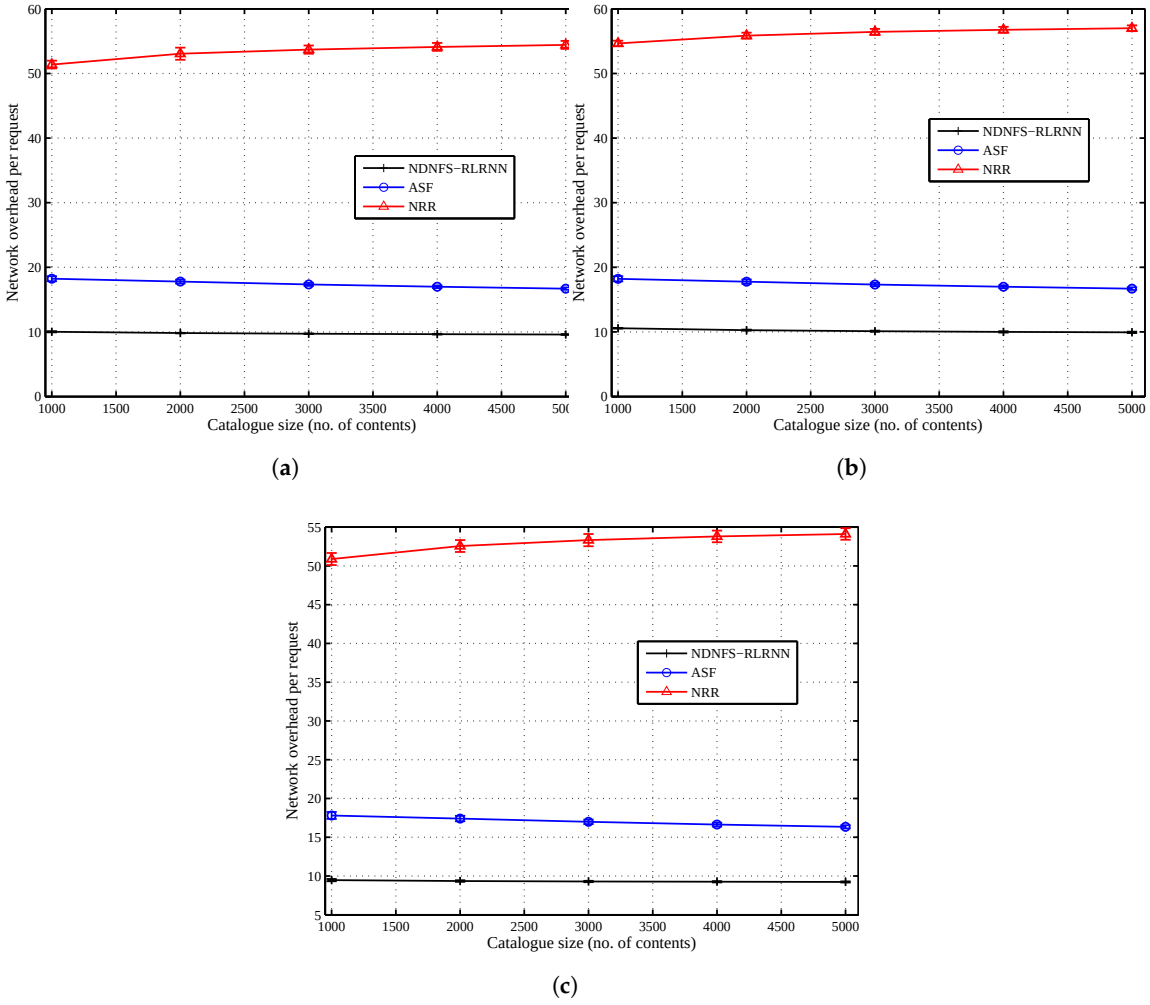


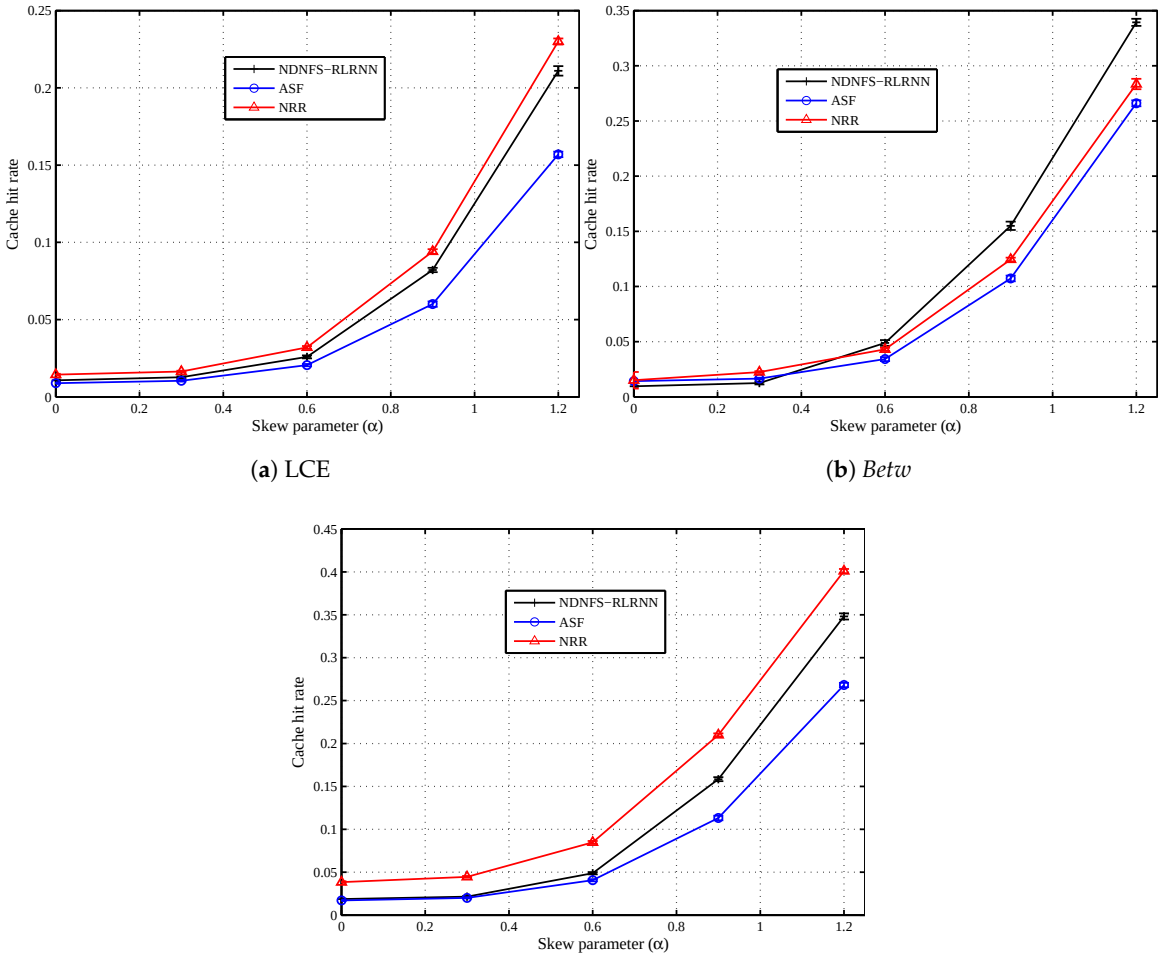
Figure 5. Network load performance of NDNFS-RLRNN ($p = 0.3$), ASF and NRR for different cache size to catalogue size ratios, under: (**a**) LCE caching policy, (**b**) *Betw* caching policy and (**c**) *ProbCache* caching policy. The cache size of each CR is fixed at 10 data objects while we vary the catalogue size from 1000 to 5000 data objects. The cache size of each CR is fixed at 10 data objects. Content popularity is modelled according to the Mandelbrot-Zipf (MZipf) distribution with the settings: $p = 5.0, \alpha \in [0.6, 1.2]$. The results reported are the average of 20 randomized simulation runs.

6.3. Content Popularity

In Figures 6 and 7, we compare the three forwarding strategies and show the impact of the content popularity model. The request distribution is uniform in the network, and we consider 5 different values of the skew parameter ($\alpha = 0.0, 0.3, 0.6, 0.9, 1.2$), including $\alpha = 0$ which corresponds to equally popular items. The closer the skew parameter is to 0, the more ineffective in-network caching will be for the same catalogue size. Our observations confirm this because as most of the requests concentrate on a smaller proportion of the content catalogue, that is, as α increases, all the forwarding strategies can hit more local caches. Also for $\alpha = 0.0$ and $\alpha = 0.3$, both ASF and NDNFS-RLRNN deliver almost identical performance with respect to the cache hit rate under the different caching policies. In fact, under the *Betw* policy, ASF is slightly better than NDNFS-RLRNN for these values of α in terms of the cache hit rate. However, as α is increased from 0.6 to 1.2, NDNFS-RLRNN produces a performance improvement of between 27–43% compared with ASF.

Finally, our results suggest that for ubiquitous caching policies like LCE and *ProbCache*, increasing the exploration in our proposed forwarding strategy, can deliver improved results with respect to exploiting in-network caching, at the cost of higher overhead as shown by the effect of increasing

the probing parameter in Figure 3. While the above statement is also true for the *Betw* policy, interestingly, it is only under this policy that the NDNFS-RLRNN performs better than the NRR despite using significantly lower probing. Since *Betw* mostly caches in specific nodes in the network, this demonstrates NDNFS-RLRNN's intelligence to search in the most worthwhile areas more efficiently than an approach that searches the entire network.



(**a**) LCE (**b**) *Betw*

(**c**) *ProbCache*

Figure 6. Cache hit rate performance of NDNFS-RLRNN ($p = 0.3$), ASF and NRR for different uniform content popularity distributions, under: (**a**) LCE caching policy, (**b**) *Betw* caching policy and (**c**) *ProbCache* caching policy. Content popularity is modelled according to the Mandelbrot-Zipf (MZipf) distribution with the settings: $p = 5.0, \alpha = 0.0, 0.3, 0.6, 0.91.2$. Each CR is equipped with a cache size of 0.4% of the content catalogue. The results reported are the average of 20 randomized simulation runs.

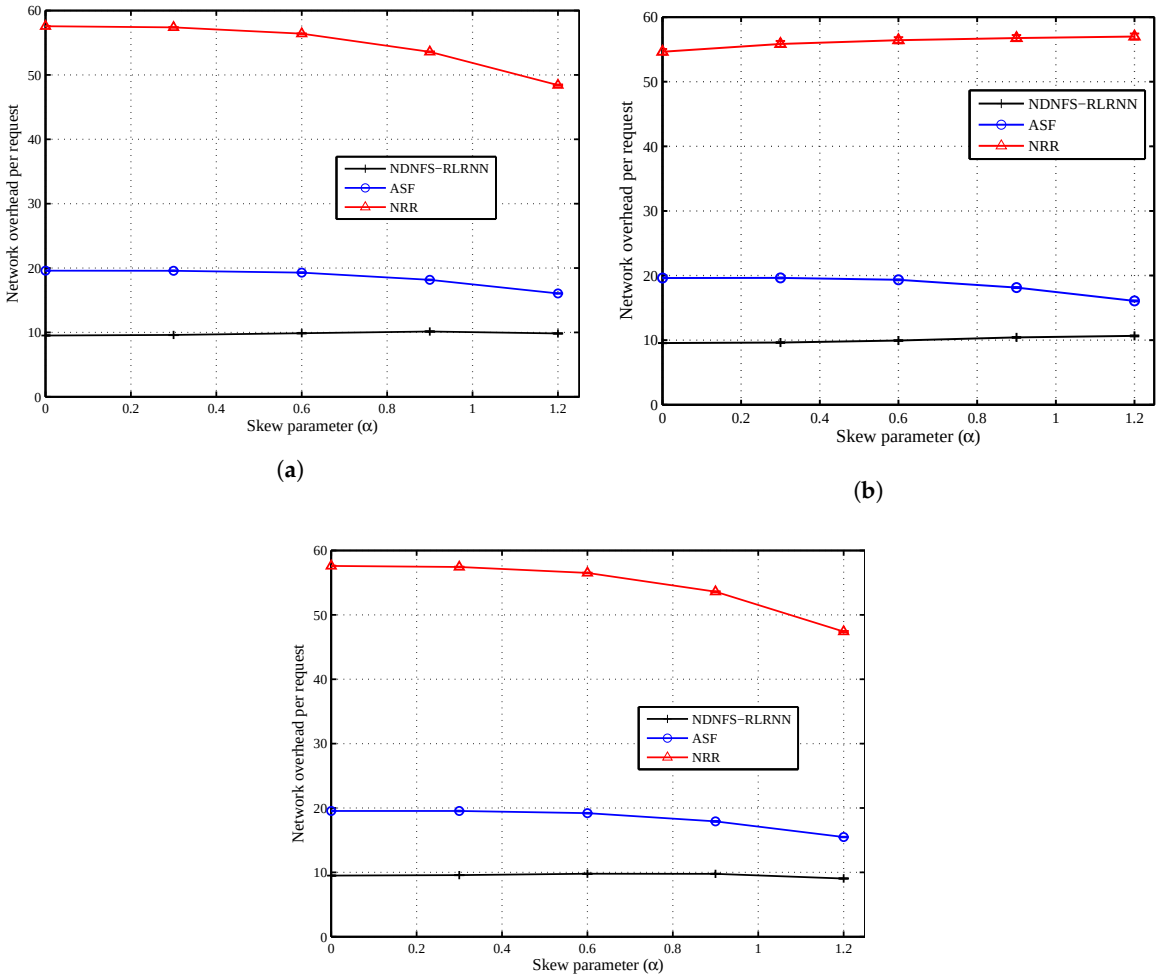


Figure 7. Network load performance of NDNFS-RLRNN ($p = 0.3$), ASF and NRR for different uniform content popularity distributions, under: (**a**) LCE caching policy, (**b**) *Betw* caching policy and (**c**) *ProbCache* caching policy. Content popularity is modelled according to the Mandelbrot-Zipf (MZipf) distribution with the settings: $p = 5.0, \alpha = 0.0, 0.3, 0.6, 0.91.2$. Each CR is equipped with a cache size of 0.4% of the content catalogue. The results reported are the average of 20 randomized simulation runs.

7. Conclusions and Further Work

This paper has addressed Information Centric Networks in the framework of Named Data Networking (NDN). We have proposed an adaptive forwarding strategy, NDNFS-RLRNN for the NDN architecture which employs an online learning algorithm, reinforcement learning using the random neural network, to forward Interest packets. Our proposed approach is dynamic and does not persist on the links put forward by the routing protocol, so that it may better recognize the role of the forwarding plane to take advantage of the in-network caching capability of the NDN architecture.

In our tests, we compare NDNFS-RLRNN with two other forwarding strategies: one that restricts forwarding and probing of Interests to the interfaces suggested by the routing protocol and another that broadcasts each Interest. We also considered three different caching policies from the literature to evaluate the forwarding strategies under different caching behaviours. We consider two ubiquitous caching policies exhibiting different cache eviction rate properties and a policy that bases its caching decisions on the network topological information.

Our results suggest that when in-network caching is effective, NDNFS-RLRNN strategy can achieve better delivery than a forwarding strategy that persists on the existing static routing layer preferences and a more efficient performance than a nearest replica forwarding strategy that floods requests. Furthermore, the results also indicate that for ubiquitous caching policies, NDNFS-RLRNN's performance can be significantly improved by increasing its exploration of the network. For a policy that caches contents at specific nodes based on the network topology, we see that the rate of improvement in terms of exploiting in-network caching reduces as the probing frequency of NDNFS-RLRNN increases. However, the results also show that even with a low probing frequency, NDNFS-RLRNN produces the best performance compared with the other forwarding strategies under this policy.

Future work will focus on evaluating our approach for congested systems, and also in the presence of denial of service attacks. Such evaluations can also lead to improvements in the adaptive on-line policy and result in a better evaluation of the computational overhead of our adaptive on-line approach.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cisco, V. Cisco Visual Networking Index: Forecast and Methodology 2016–2021. 2017. Available online: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html (accessed on 30 June 2018).
2. Wendell, P.; Freedman, M.J. Going Viral: Flash Crowds in an Open CDN. In Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, New York, NY, USA, 2–4 November 2011; pp. 549–558.
3. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A Survey of Information-Centric Networking Research. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1024–1049. [CrossRef]
4. Ahlgren, B.; Dannewitz, C.; Imbrenda, C.; Kutscher, D.; Ohlman, B. A survey of information-centric networking. *IEEE Commun. Mag.* **2012**, *50*, 26–36. [CrossRef]
5. NSF Named Data Networking Project. Available online: https://named-data.net (accessed on 20 July 2018).
6. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, K.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named Data Networking. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
7. Lehman, V.; Gawande, A.; Zhang, B.; Zhang, L.; Aldecoa, R.; Krioukov, D.; Wang, L. An experimental investigation of hyperbolic routing with a smart forwarding plane in NDN. In Proceedings of the IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), 20–21 June 2016; pp. 1–10.
8. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking Named Content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, New York, NY, USA, 1–4 December 2009; pp. 1–12.
9. Afanasyev, A.; Shi, J.; Zhang, B.; Zhang, L.; Moiseenko, I.; Yu, Y.; Shang, W.; Huang, Y.; Abraham, J.P.; DiBenedetto, S.; et al. *NFD Developer's Guide*; Technical Report NDN-0021; Department of Computer Science, University of California: Los Angeles, CA, USA, 2014.
10. Yi, C.; Afanasyev, A.; Wang, L.; Zhang, B.; Zhang, L. Adaptive Forwarding in Named Data Networking. *SIGCOMM Comput. Commun. Rev.* **2012**, *42*, 62–67. [CrossRef]
11. Garcia-Luna-Aceves, J.; Mirzazad-Barijough, M. Enabling Correct Interest Forwarding and Retransmissions in a Content Centric Network. In Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Oakland, CA, USA, 7–8 May 2015; pp. 135–146.
12. Gelenbe, E. Self-aware Networks: The Cognitive Packet Network and Its Performance. In *Self-Aware Computing Systems*; Springer: Berlin, Germany, 2017; pp. 659–668.
13. Gelenbe, E. Réseaux neuronaux aléatoires stables. *Comptes-Rendus de l'Académie des Sciences* **1990**, *310*, 177–180.
14. Gelenbe, E. Steps Toward Self-aware Networks. *Commun. ACM* **2009**, *52*, 66–75. [CrossRef]

15. Birke, R.; Cámara, J.; Chen, L.Y.; Esterle, L.; Geihs, K.; Gelenbe, E.; Giese, H.; Robertsson, A.; Zhu, X. Self-aware Computing Systems: Open Challenges and Future Research Directions. In *Self-Aware Computing Systems*; Springer: Berlin, Germany, 2017; pp. 709–722.
16. Brun, O.; Wang, L.; Gelenbe, E. Big Data for Autonomic Intercontinental Overlays. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 575–583. [CrossRef]
17. Halici, U. Reinforcement learning with internal expectation for the random neural network. *Eur. J. Oper. Res.* **2000**, *126*, 288–307. [CrossRef]
18. Bari, M.F.; Chowdhury, S.R.; Ahmed, R.; Boutaba, R.; Mathieu, B. A survey of naming and routing in information-centric networks. *IEEE Commun. Mag.* **2012**, *50*, 44–53. [CrossRef]
19. Rossini, G.; Rossi, D. Coupling Caching and Forwarding: Benefits, Analysis, and Implementation. In Proceedings of the 1st ACM Conference on Information-Centric Networking, Paris, France, 24–26 September 2014; pp. 127–136.
20. Zhang, M.; Luo, H.; Zhang, H. A Survey of Caching Mechanisms in Information-Centric Networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1473–1499. [CrossRef]
21. Yi, C.; Afanasyev, A.; Moiseenko, I.; Wang, L.; Zhang, B.; Zhang, L. A Case for Stateful Forwarding Plane. *Comput. Commun.* **2013**, *36*, 779–791. [CrossRef]
22. Qian, H.; Ravindran, R.; Wang, G.Q.; Medhi, D. Probability-based adaptive forwarding strategy in named data networking. In Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium, 27–31 May 2013; pp. 1094–1101.
23. Nguyen, D.; Fukushima, M.; Sugiyama, K.; Tagami, A. Efficient multipath forwarding and congestion control without route-labeling in CCN. In Proceedings of the 2015 IEEE International Conference on Communication Workshop (ICCW), London, UK, 8–12 June 2015; pp. 1533–1538.
24. Posch, D.; Rainer, B.; Hellwagner, H. SAF: Stochastic Adaptive Forwarding in Named Data Networking. *IEEE/ACM Trans. Netw.* **2017**, *25*, 1089–1102. [CrossRef]
25. Bastos, I.V.; Moraes, I.M. A forwarding strategy based on reinforcement learning for Content-Centric Networking. In Proceedings of the 7th International Conference on the Network of the Future (NOF), Rio de Janeiro, Brazil, 16–18 November 2016; pp. 1–5.
26. Chiocchetti, R.; Perino, D.; Carofiglio, G.; Rossi, D.; Rossini, G. INFORM: A Dynamic Interest Forwarding Mechanism for Information Centric Networking. In Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking, Hong Kong, China, 12 August 2013; pp. 9–14.
27. Boyan, J.A.; Littman, M.L. Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach. In Proceedings of the 6th International Conference on Neural Information Processing Systems, San Francisco, CA, USA, 26–28 July 1993; pp. 671–678.
28. Sakellari, G. The Cognitive Packet Network: A Survey. *Comput. J.* **2010**, *53*, 268. [CrossRef]
29. Gelenbe, E.; Pujolle, G. *Introduction to Networks of Queues*; John Wiley Ltd.: New York, NY, USA, 1998.
30. Gelenbe, E.; Sungur, M.; Cramer, C.; Gelenbe, P. Traffic and Video Quality with Adaptive Neural Compression. *Multimedia Syst.* **1996**, *4*, 357–369. [CrossRef]
31. Cramer, C.; Gelenbe, E. Video quality and traffic QoS in learning-based subsampled and receiver-interpolated video sequences. *IEEE J. Sel. Areas Commun.* **2000**, *18*, 150–167. [CrossRef]
32. Gelenbe, E.; Feng, Y.; Krishnan, K.R.R. Neural network methods for volumetric magnetic resonance imaging of the human brain. *Proc. IEEE* **1996**, *84*, 1488–1496. [CrossRef]
33. Grenet, I.; Yin, Y.; Comet, J.P.; Gelenbe, E. Machine Learning to Predict Toxicity of Compounds. In Proceedings of the 27th Annual International Conference on Artificial Neural Networks, Rhodes, Greece, 4–7 October 2018.
34. Serrano, W.; Gelenbe, E. The Random Neural Network in a neurocomputing application for Web search. *Neurocomputing* **2018**, *280*, 123–134. [CrossRef]
35. Gelenbe, S.E. Cognitive Packet Network. U.S. Patent 6,804,201, 5 October 2004.
36. Gelenbe, E.; Lent, R. Power-aware ad hoc cognitive packet networks. *Ad Hoc Netw.* **2004**, *2*, 205–216. [CrossRef]
37. Gelenbe, E.; Liu, P.; LainLaine, J. Genetic algorithms for route discovery. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2006**, *36*, 1247–1254. [CrossRef]

38. François, F.; Gelenbe, E. Optimizing Secure SDN-Enabled Inter-Data Centre Overlay Networks through Cognitive Routing. In Proceedings of the MASCOTS 2016, IEEE Computer Society, London, UK, 19–21 September 2016; pp. 283–288.
39. Wang, L.; Gelenbe, E. Adaptive dispatching of tasks in the cloud. *IEEE Trans. Cloud Comput.* **2018**, *6*, 33–45. [CrossRef]
40. Gelenbe, E. Product-form queueing networks with negative and positive customers. *J. Appl. Probab.* **1991**, *28*, 656–663. [CrossRef]
41. Gelenbe, E.; Glynn, P.; Sigman, K. Queues with negative arrivals. *J. Appl. Probab.* **1991**, *28*, 245–250. [CrossRef]
42. Gelenbe, E.; Schassberger, R. Stability of product form G-networks. *Probab. Eng. Inf. Sci.* **1992**, *6*, 271–276. [CrossRef]
43. Gelenbe, E. G-networks by triggered customer movement. *J. Appl. Probab.* **1993**, *30*, 742–748. [CrossRef]
44. Gelenbe, E. G-networks with signals and batch removal. *Probab. Eng. Inf. Sci.* **1993**, *7*, 335–342. [CrossRef]
45. Fourneau, J.M.; Gelenbe, E. G-networks with adders. *Future Internet* **2017**, *9*, 34. [CrossRef]
46. Gelenbe, E. Random Neural Networks with Negative and Positive Signals and Product Form Solution. *Neural Comput.* **1989**, *1*, 502–510. [CrossRef]
47. Gelenbe, E. Learning in the recurrent random neural network. *Neural Comput.* **1993**, *5*, 154–164. [CrossRef]
48. Görbil, G.; Gelenbe, E. Opportunistic communications for emergency support systems. *Procedia Comput. Sci.* **2011**, *5*, 39–47. [CrossRef]
49. Desmet, A.; Gelenbe, E. A Parametric Study of CPN's Convergence Process. In *Information Sciences and Systems 2014*; Czachórski, T., Gelenbe, E., Lent, R., Eds.; Springer: Cham, Switzerland, 2014; pp. 13–20.
50. Gelenbe, E.; Kazhmaganbetova, Z. Cognitive packet network for bilateral asymmetric connections. *IEEE Trans. Ind. Inf.* **2014**, *10*, 1717–1725. [CrossRef]
51. Coffman, E.G., Jr.; Denning, P.J. *Operating Systems Theory*; Prentice Hall Professional Technical Reference: Upper Saddle River, NJ, USA, 1973.
52. Che, H.; Tung, Y.; Wang, Z. Hierarchical Web caching systems: Modeling, design and experimental results. *IEEE J. Sel. Areas Commun.* **2002**, *20*, 1305–1314. [CrossRef]
53. Dan, A.; Towsley, D. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. *SIGMETRICS Perform. Eval. Rev.* **1990**, *18*, 143–152. [CrossRef]
54. Gelenbe, E. A Unified Approach to the Evaluation of a Class of Replacement Algorithms. *IEEE Trans. Comput.* **1973**, *C-22*, 611–618. [CrossRef]
55. Traverso, S.; Ahmed, M.; Garetto, M.; Giaccone, P.; Leonardi, E.; Niccolini, S. Temporal Locality in Today's Content Caching: Why It Matters and How to Model It. *SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 5–12. [CrossRef]
56. Almeida, V.; Bestavros, A.; Crovella, M.; de Oliveira, A. Characterizing reference locality in the WWW. In Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems, Miami Beach, FL, USA, 18–20 December 1996; pp. 92–103.
57. Breslau, L.; Cao, P.; Fan, L.; Phillips, G.; Shenker, S. Web caching and Zipf-like distributions: Evidence and implications. In Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99), New York, NY, USA, 21–25 March 1999; pp. 126–134
58. Jin, S.; Bestavros, A. Sources and characteristics of Web temporal locality. In Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (Cat. No. PR00728), San Francisco, CA, USA, 29 August 2000; pp. 28–35.
59. Muscariello, L.; Carofiglio, G.; Gallo, M. Bandwidth and Storage Sharing Performance in Information Centric Networking. In Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking, Toronto, ON, Canada, 19 August 2011; pp. 26–31.
60. Carofiglio, G.; Gallo, M.; Muscariello, L.; Perino, D. Modeling data transfer in content-centric networking. In Proceedings of the 23rd International Teletraffic Congress (ITC), San Francisco, CA, USA, 6–9 September 2011; pp. 111–118.
61. Rossi, D.; Rossini, G. Caching performance of content centric networks under multi-path routing (and more). *Relatório Técnico Telecom ParisTech* **2011**, 1–6. Available online: https://pdfs.semanticscholar.org/8fcc/e9e4865a950723f93bb97b5d5aa7e793037a.pdf (accessed on 20 July 2018).

62. Katsaros, K.; Xylomenos, G.; Polyzos, G.C. MultiCache: An Overlay Architecture for Information-Centric Networking. *Comput. Netw.* **2011**, *55*, 936–947. [CrossRef]
63. Fayazbakhsh, S.K.; Lin, Y.; Tootoonchian, A.; Ghodsi, A.; Koponen, T.; Maggs, B.; Ng, K.; Sekar, V.; Shenker, S. Less Pain, Most of the Gain: Incrementally Deployable ICN. *SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 147–158. [CrossRef]
64. Cha, M.; Kwak, H.; Rodriguez, P.; Ahn, Y.Y.; Moon, S. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, San Diego, CA, USA, 23–26 October 2007; pp. 1–14.
65. Hefeeda, M.; Saleh, O. Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems. *IEEE/ACM Trans. Netw.* **2008**, *16*, 1447–1460. [CrossRef]
66. Sourlas, V.; Gkatzikis, L.; Flegkas, P.; Tassiulas, L. Distributed Cache Management in Information-Centric Networks. *IEEE Trans. Netw. Serv. Manag.* **2013**, *10*, 286–299. [CrossRef]
67. Rosensweig, E.J.; Kurose, J. Breadcrumbs: Efficient, Best-Effort Content Location in Cache Networks. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), 24 July 2009; pp. 2631–2635.
68. Mastorakis, S.; Afanasyev, A.; Moiseenko, I.; Zhang, L. *ndnSIM 2: An Updated NDN Simulator for NS-3*; Technical Report NDN-0028. Available online: https://named-data.net/publications/techreports/ndn-0028-2-ndnsim-v2/ (accessed on 11 November 2016).
69. Knight, S.; Nguyen, H.; Falkner, N.; Bowden, R.; Roughan, M. The Internet Topology Zoo. *IEEE J. Sel. Areas Commun.* **2011**, *29*, 1765–1775. [CrossRef]
70. Chai, W.K.; He, D.; Psaras, I.; Pavlou, G. Cache "less for more" in information-centric networks (extended version). *Comput. Commun.* **2013**, *36*, 758–770. [CrossRef]
71. Psaras, I.; Chai, W.K.; Pavlou, G. Probabilistic In-Network Caching for Information-Centric Networks. In Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking, New York, NY, USA, 17 August 2012; pp. 55–60.