Adaptive 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

Erol Gelenbe
Department of Electrical and Electronic Engineering
Imperial College
London SW7 2AZ, UK
e.gelenbe@imperial.ac.uk

***Abstract*—This paper addresses Information-Centric Networks, and considers in-network caching specifically for Named Data Networking (NDN) architectures, and departs from 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 indicate that NDNFS-RLRNN achieves better delivery performance and lower overhead, than a strategy that uses fixed paths from the routing layer.**

***Index Terms*—Information Centric Networks, NDN, Cognitive Packet Networks, Random Neural Networks**

I. Introduction

Information-Centric Networking (ICN) treats content as fundamental by naming and addressing data objects at the network level, which is a significant departure from the Internet's host-centic architecture. This enables ICN to naturally support features like in-network caching and many-to-many communication which can be essential in meeting the current predominant use of the Internet [1], [2].

This paper focuses on *Named Data Networking* (NDN) [3], one of the prominent ICN approaches. In NDN, requests are sent in *Interest packets*, while *Data packets* carry the requested contents; both packets carry 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. An NDN node maintains three main data structures for implementing the forwarding plane: a *content store* (CS) which acts as a cache for Data packets, a *pending interest table* (PIT) which keeps track of unanswered requests, and a *forwarding information base* (FIB) which maps reachable name prefixes to outgoing interfaces. The FIB is updated by a name-based routing protocol.

The NDN architecture differs significantly from most of the other ICN proposals because it incorporates an intelligent forwarding plane through its *strategy module*. The strategy module is a program defined for each prefix in the FIB, which makes forwarding decisions for Interest forwarding. It is expected to leverage on both the routing information in the FIB and the observed packet delivery measurements in making adaptive decisions. An important benefit of this layer is that it can relax the stringent convergence and correctness demands on the routing layer. An NDN node matches an Interest, using the content name, in the CS, PIT and FIB, in that order. A match in the FIB results in the strategy being called upon to decide the Interest's next hop. Data packets are sent along the reverse paths used by their corresponding Interest packets, following the states in the PITs.

For scalability and practicality reasons, the name-based routing protocol in NDN can only announce and monitor paths leading to the actual content sources and designated repositories, thus leaving the responsibility of exploiting the in-network caching capability, which is important for delivering the design goals of NDN, entirely to the forwarding strategy. However, most of the proposals for the strategy layer, including the algorithm currently adopted by the NDN project, follow a "monitor-the-routing-paths" approach, whereby their forwarding and probing actions are restricted to the links suggested by the routing layer [4]–[7]. This approach limits the strategy from exploiting local caches closer but outside the routing paths.

Here, we develop a dynamic self-aware [8] 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 strategy module that we implement exploits the Random Neural Network (RNN) [9] with Reinforcement Learning, similar to the Cognitive Packet Network (CPN) [10], [11]. An overlay network with a similar scheme is described in [12].

The next section reviews previous work, while Section III presents a reinforcement learning algorithm for the RNN [13]. The *NDN forwarding strategy using reinforcement learning with the RNN* (NDNFS-RLRNN) is detailed in Section IV, and its performance evaluation is reported in Section V. Conclusions and future work are discussed in Section VI.

II. Related Work

Simple strategies have been proposed for and tested in NDN. Forwarding an Interest along all the interfaces suggested by the routing layer excluding the interface the Interest arrives on, also called the *multicast strategy* was proposed in [4]. The *best route strategy* [5] forwards interests using

the available upstream with the lowest routing costs. The current NDN strategy [3], [6], [14] combines interface ranking and colour classification in order 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 SRTT. The forwarding logic is to use the highest ranked available interface in the best possible classification. Interest NACKs were introduced in NDN 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, an NDN router 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) was proposed in [7]. In SIFAH, a node accepts to forward an Interest packet only if there exists, based on distance information to the content source, a neighbour node that moves the packet closer to the content. 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. 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.

There is also a class of strategies, referred to as the multipath forwarding strategies [15]–[17], that dynamically assign each interface of an NDN router a forwarding weight or probability per name prefix, which determines the proportion of request traffic sent on each interface. The main aim here is to achieve good 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 [18] the *multi-armed bandits strategy* (MABS) is developed that assumes no knowledge of path information from the routing layer. Rather than leverage on routing information, requests are flooded when no learning has occurred. *INFORM* presented in [19] and inspired by the Q-routing algorithm [20], 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 is the one learnt by the Q-learning 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 possible inefficiencies that could be introduced by the dangling states in the PIT and the handling of Interest NACKs.

TABLE I
NOTATION FOR THE RNN MODEL

Notation	Definition
$k(t)$	State vector of the RNN at time t where $k_i(t) >= 0$ 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

Our approach is inspired from the CPN routing protocol which, in most of its implementations, employs a reinforcement learning algorithm using the RNN 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 [13], hence its success in CPN routing. A comprehensive review of the variations, applications, and performance evaluations of the CPN can be found in [21].

III. REINFORCEMENT LEARNING AND THE RANDOM NEURAL NETWORK

The RNN [22] is a neural network model where neurons send and receive both positive and negative signals, which has been extensively used for network routing and Cloud management [23]–[27]. The positive signals are called *excitatory*, while the negative signals are called *inhibitory*.

Table I defines important notations for the RNN model.

Since there are no self-loop connections in the network, we can write for each neuron i in an n-neuron network

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

where $p(i,j)=p^+(i,j)+p^-(i,j)$. It was shown in [28] that the RNN model with exponential firing intervals and Poisson exogenous signal arrivals has the product form soulution

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

$$q_i=\frac{\lambda_i^+}{(r_i+\lambda_i^-)}, \tag{3}$$

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^- must satisfy the following system of non-linear equations

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

$$\lambda_i^- = \sum_j [q_j r_j p^-(j,i)] + \lambda_i \tag{5}$$

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

$$r_i = \sum_{i=1}^{n} [w^+(i,j) + w^-(i,j)] \quad i = 1, 2, ..., n \tag{6}$$

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 weights being equal. The pseudocode in Algorithm 1 illustrates how reinforcement learning can be implemented using the RNN. The reinforcement learning algorithm used is based on the *E-rule* scheme in [13] which assumes that a reward value can be estimated for every decision. In this approach, the outcome of a previous decision is compared with an internal expectation 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. 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. This is shown in lines $1 - 12$ of Algorithm 1. At any given time, the neuron with the highest excitation probability q is considered the best decision. Line 21 shows how the excitation probabilities are estimated after the weights are updated.

IV. NDN Forwarding with Reinforcement Learning using the RNN

We present NDNFS-RLRNN that uses reinforcement learning with the RNN for the strategy module per prefix in the NDN architecture, which operates with a form of smart Opportunistic Communication [29], supported by a name-based routing protocol with online measurements from the state recorded in the PIT. An RNN is created for each prefix announced into the FIB by the routing protocol. An RNN in the initial state only knows of the routing preferences for its prefix and is yet to be updated by packet delivery measurements. In this state, NDNFS-RLRNN's forwarding decisions will use the best interface according to the routing layer. The RNN at the NDN router has as many neurons as the number k of outgoing links of the router, and the RNN has k^2 wights, 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 $n_D.k^2$ entries. A conventional routing table at the NDN router will have $n_S.n_D.k$ 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 [30].

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

$$R^j = ({T^j}_D - {T^j}_I)^{-1} \tag{7}$$

In applying the reinforcement learning algorithm, it is important to consider in this application that the links suggested by the routing protocol offer better assurances on content delivery. Therefore, only positive reinforcements are applied to the neurons corresponding to these interfaces, ensuring that alternative interfaces will be used only when they are known to improve delivery. Also, whenever an entry in the PIT expires without any response or the RNN receives no updates for T_r time units, the learning algorithm reinitializes, thereby falling back to the routing layer preferences.

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. 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 NDN routers 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 an NDN router 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 NDNFS-RLRNN, we have also adopted the use of NACKS as in the current NDN. Clearly, the reward computation is explicit when the Interest is satisfied; but, this is not the case when a NACK is returned or no response is received

before the Interest times out. In such cases, the reward can be estimated as

$$R^j = [c(T - T^j{}_I)]^{-1} \quad (8)$$

where T is the current time and c is a constant large enough such that a negative reinforcement is applied.

V. Performance Evaluation

In this section, we present initial evaluations of the performance of NDNFS-RLRNN through extensive simulations using the *ndnSim* [31], 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 1 according to the dataset in [32]. Each node receives external requests for

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 $R^i{}_l >= T_{l-1}$ **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 (3) for each $0 <= q_j <= 1$ */
/* 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 */

contents at an average of 5 request packets per second with a Poisson distribution. We model the requests according to the Mandelbrot-Zipf (MZipf) distribution. The plateau parameter for the distribution is fixed at $p = 5.0$ while the skew

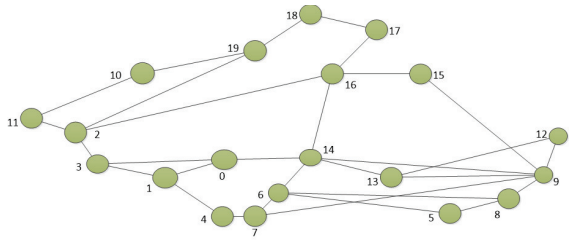

Fig. 1. *Elibackbone* topology. The network consists of 20 nodes and 30 links.

parameter, α at each node is randomly chosen in the range [0.6, 1.2]. The cache capacity is fixed for each node at 10 data units and we investigate network performance [33] under different catalogue sizes. The simulations begin with empty caches at the nodes. We consider a single access router for the network through which requests are served from the content servers.

Furthermore, we compare NDNFS-RLRNN with the *Adaptive SRTT-based Forwarding Strategy* (ASF) [34] that uses the upstream with the lowest measured SRTT and probes alternative interfaces suggested in the FIB at intervals. The length of a probing interval is chosen to be 3 seconds and we install all possible routes in the FIB such that no loops exist in the forwarding. We have considered a small probing intervals for ASF to compare with a probing probability of 0.3 for NDNFS-RLRNN. The strategies are implemented per content in the catalog.

Also, since the cache states in the network will influence the performances of the strategies, we compare them under 3 different cache policies from the literature:

- Leave copy everywhere combined with LRU replacement policy (*LCE*).
- Betweenness centrality policy and LRU (*Betw*): In *Betw* [35], only the routers with the highest betweenness centrality values along the delivery path cache the content.
- *ProbCache and LRU*: Here routers cache content based on some set probability. *ProbCache* [36] computes a router's caching probability by weighing the caching capacity of the delivery path with the relative position of the router along the path.

Finally, we measure and report the *cache hit rate* and the *network load per request*. The cache hit rate only considers the hits that actually satisfy the user requests; it is 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 evaluated as the total number of hops traversed by the network packets, which includes Interests, Data and NACK packets, per request sent into the network.

A. Results

We present some resuts in Figures 2 - 5. Each plot on the graph represents the average value from 5 randomised simulation runs which last for 400 seconds and the error bars represent the standard deviations. Fig. 2, 3 and 4 compare

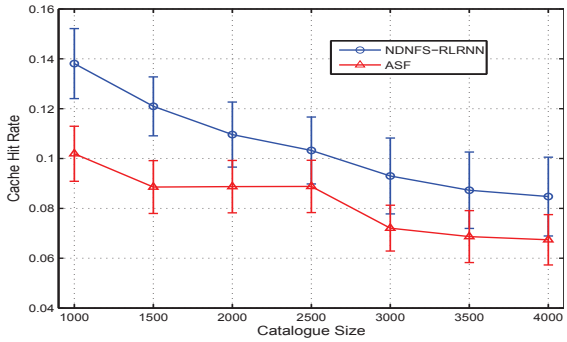

Fig. 2. Cache hit rate performance of NDNFS-RLRNN and ASF under the *LCE* and *LRU* cache policy for different catalogue sizes. The cache size of each node is fixed at 10 data objects. Content popularity is modelled according to the Mandelbrot-Zipf (MZipf) distribution with the settings, $p = 5.0, \alpha \in [0.6, 1.2]$. The results reported are the average of 5 randomized simulation runs.

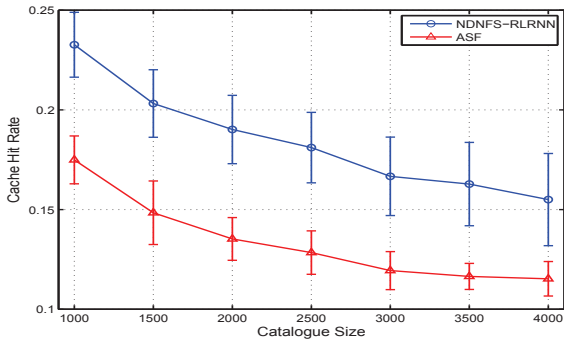

Fig. 3. Cache hit rate performance of NDNFS-RLRNN and ASF under the *Betw* and *LRU* cache policy for different catalogue sizes. The cache size of each node is fixed at 10 data objects. Content popularity is modelled according to the Mandelbrot-Zipf (MZipf) distribution with the settings, $p = 5.0, \alpha \in [0.6, 1.2]$. The results reported are the average of 5 randomized simulation runs.

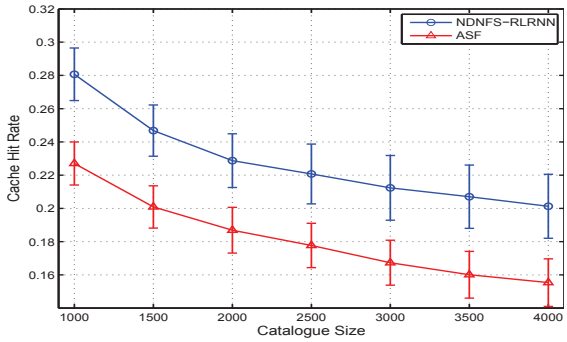

Fig. 4. Cache hit rate performance of NDNFS-RLRNN and ASF under the *ProbCache* and *LRU* cache policy for different catalogue sizes. The cache size of each node is fixed at 10 data objects. Content popularity is modelled according to the Mandelbrot-Zipf (MZipf) distribution with the settings, $p = 5.0, \alpha \in [0.6, 1.2]$. The results reported are the average of 5 randomized simulation runs.

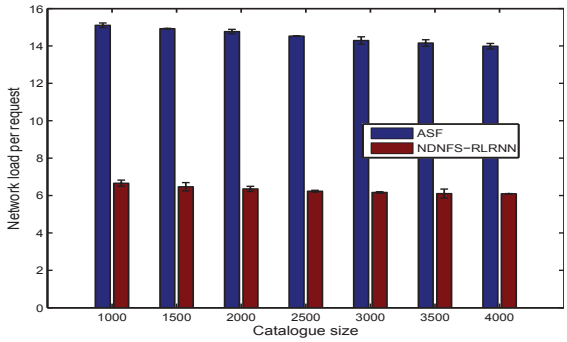

Fig. 5. Interest load performance of NDNFS-RLRNN and ASF under the *ProbCache* and *LRU* cache policy for different catalogue sizes. The cache size of each node is fixed at 10 data objects. Content popularity is modelled according to the Mandelbrot-Zipf (MZipf) distribution with the settings, $p = 5.0, \alpha \in [0.6, 1.2]$. The results reported are the average of 5 randomized simulation runs.

the cache hit performances of the considered strategies under 3 cache policies. We see that NDNFS-RLRNN delivers a better performance than ASF under the considered cache policies. Fig. 2 shows the hit rates achieved by the strategies when the LCE policy is used at the NDN routers. Despite the characteristic high redundancy and high eviction rate of LCE, NDNFS-RLRNN still exploits the local caches better than ASF which is restricted to the delivery paths according to the FIB. Although, as the catalogue size increases and caching becomes less effective, the improvement over ASF reduces. Fig. 3 shows improved results generally as the cache policy tries to cache packets at the most "central" nodes along the delivery path. While *Betw* policy has reduced redundancy compared with the LCE policy, the replacement rates will be high at the most central nodes. Under these conditions, compared with ASF, NDNFS-RLRNN delivers an improvement between $31 - 42\%$ with respect to the cache hit rate. The *ProbCache* produces the best hit results for the strategies as observed in Fig. 4 because it combines better redundancy and eviction properties . Here, NDNFS-RLRNN provides an average performance improvement of about 31% over ASF for the considered catalogue sizes. These results suggest that when caching is effective, NDNFS-RLRNN can take better advantage compared with ASF.

Finally, due to limited space, we only report the overhead results under the ProbCache policy in Fig. 5. Fig. 5 shows that NDNFS-RLRNN delivers the improvements in the hit rates with significantly lower overhead; ASF incurs more than double the overhead measured for NDNFS-RLRNN.

VI. 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 on-line algorithm in order 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.

Our experiments suggest that when in-network caching is effective, NDNFS-RLRNN strategy can achieve better delivery than a strategy that persists on the existing static routing layer preferences. Future work will focus on evaluating our approach for congested systems, and also in the presence 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.

REFERENCES

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

[2] V. Cisco, "Cisco visual networking index: Forecast and methodology 2016–2021.(2017)," 2017.

[3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2656877.2656887

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658941

[5] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto *et al.*, "Nfd developers guide," *Dept. Comput. Sci., Univ. California, Los Angeles, Los Angeles, CA, USA, Tech. Rep. NDN-0021*, 2014.

[6] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 62–67, Jun. 2012. [Online]. Available: http://doi.acm.org/10.1145/2317307.2317319

[7] J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, "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*, ser. ANCS '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 135–146. [Online]. Available: http://dl.acm.org/citation.cfm?id=2772722.2772741

[8] E. Gelenbe, "Self-aware networks: The cognitive packet network and its performance," in *Self-Aware Computing Systems*. Springer, 2017, pp. 659–668. [Online]. Available: https://doi.org/10.1007/978-3-319-47474-8

[9] E. Gelenbe and J. M. Fourneau, "Random neural networks with multiple classes of signals," *Neural Computation*, vol. 11, no. 4, pp. 953–963, May 1999.

[10] E. Gelenbe, "Steps toward self-aware networks," *Commun. ACM*, vol. 52, no. 7, pp. 66–75, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1538788.1538809

[11] R. Birke, J. Cámara, L. Y. Chen, L. Esterle, K. Geihs, E. Gelenbe, H. Giese, A. Robertsson, and X. Zhu, "Self-aware computing systems: Open challenges and future research directions," in *Self-Aware Computing Systems*. Springer, 2017, pp. 709–722. [Online]. Available: https://doi.org/10.1007/978-3-319-47474-8_26

[12] O. Brun, L. Wang, and E. Gelenbe, "Big data for autonomic intercontinental overlays," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 575–583, March 2016.

[13] U. Halici, "Reinforcement learning with internal expectation for the random neural network," *European Journal of Operational Research*, vol. 126, no. 2, pp. 288 – 307, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221799004798

[14] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Comput. Commun.*, vol. 36, no. 7, pp. 779–791, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2013.01.005

[15] H. Qian, R. Ravindran, G. Q. Wang, and D. Medhi, "Probability-based adaptive forwarding strategy in named data networking," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, May 2013, pp. 1094–1101.

[16] D. Nguyen, M. Fukushima, K. Sugiyama, and A. Tagami, "Efficient multipath forwarding and congestion control without route-labeling in ccn," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, June 2015, pp. 1533–1538.

[17] D. Posch, B. Rainer, and H. Hellwagner, "SAF: Stochastic adaptive forwarding in named data networking," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1089–1102, April 2017.

[18] I. V. Bastos and I. M. Moraes, "A forwarding strategy based on reinforcement learning for content-centric networking," in *2016 7th International Conference on the Network of the Future (NOF)*, Nov 2016, pp. 1–5.

[19] R. Chiocchetti, D. Perino, G. Carofiglio, D. Rossi, and G. Rossini, "Inform: A dynamic interest forwarding mechanism for information centric networking," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '13. New York, NY, USA: ACM, 2013, pp. 9–14. [Online]. Available: http://doi.acm.org/10.1145/2491224.2491227

[20] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proceedings of the 6th International Conference on Neural Information Processing Systems*, ser. NIPS'93. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 671–678. [Online]. Available: http://dl.acm.org/citation.cfm?id=2987189.2987274

[21] G. Sakellari, "The cognitive packet network: A survey," *The Computer Journal*, vol. 53, no. 3, p. 268, 2010. [Online]. Available: +http://dx.doi.org/10.1093/comjnl/bxp053

[22] E. Gelenbe, "Réseaux neuronaux alatoires stables," *Comptes-rendus de l'Académie des Sciences. Série 2*, vol. 310, no. 3, pp. 177–180, 1990.

[23] S. E. Gelenbe, "Cognitive packet network," *US Patent*, no. 6804201, 2004.

[24] E. Gelenbe and R. Lent, "Power-aware ad hoc cognitive packet networks," *Ad Hoc Networks*, vol. 2, no. 3, pp. 205–216, 2004.

[25] E. Gelenbe, P. Liu, and J. Lainé, "Genetic algorithms for route discovery," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 6, pp. 1247–1254, 2006.

[26] F. François and E. Gelenbe, "Optimizing secure sdn-enabled inter-data centre overlay networks through cognitive routing," in *MASCOTS 2016, IEEE Computer Society*, 2018, pp. 283–288.

[27] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *IEEE Trans. Cloud Computing*, vol. 6, no. 1, pp. 33–45, 2018.

[28] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Comput.*, vol. 1, no. 4, pp. 502–510, Dec. 1989. [Online]. Available: http://dx.doi.org/10.1162/neco.1989.1.4.502

[29] G. Görbil and E. Gelenbe, "Opportunistic communications for emergency support systems," pp. 39–47, 2011.

[30] E. Gelenbe and Z. Kazhmaganbetova, "Cognitive packet network for bilateral asymmetric connections," *IEEE Trans. Industrial Informatics*, vol. 10, no. 3, pp. 1717–1725, 2014. [Online]. Available: https://doi.org/10.1109/TII.2014.2321740

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

[32] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765 –1775, october 2011.

[33] E. Gelenbe and G. Pujolle, *Introduction to networks of queues*. John Wiley Ltd., 1998.

[34] V. Lehman, A. Gawande, B. Zhang, L. Zhang, R. Aldecoa, D. Krioukov, and L. Wang, "An experimental investigation of hyperbolic routing with a smart forwarding plane in ndn," in *2016 IEEE/ACM 24th International Symposium on Quality of Service, IWQoS 2016*. USA: IEEE, 10 2016.

[35] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks (extended version)," *Computer Communications*, vol. 36, no. 7, pp. 758 – 770, 2013.

[36] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ser. ICN '12. New York, NY, USA: ACM, 2012, pp. 55–60. [Online]. Available: http://doi.acm.org/10.1145/2342488.2342501