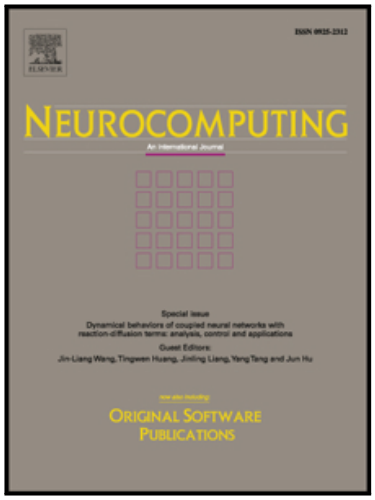
Communicated by Dr. Nianyin Zeng

Accepted Manuscript



A SMDP-based Forwarding Scheme in Named Data Networking

Jinfa Yao, Baoqun Yin, Xiaobin Tan

PII: S0925-2312(18)30447-8
DOI: 10.1016/j.neucom.2018.03.057
Reference: NEUCOM 19480

To appear in: *Neurocomputing*

Received date: 1 October 2017
Revised date: 25 February 2018
Accepted date: 29 March 2018

Please cite this article as: Jinfa Yao, Baoqun Yin, Xiaobin Tan, A SMDP-based Forwarding Scheme in Named Data Networking, *Neurocomputing* (2018), doi: 10.1016/j.neucom.2018.03.057

A SMDP-based Forwarding Scheme in Named Data Networking

Jinfa Yao*, Baoqun Yin, Xiaobin Tan

Department of Automation, University of Science and Technology of China, Hefei, 230027, China

Abstract

Named data networking (NDN) provides a promising networking paradigm for efficient content delivery. In NDN, adaptive request forwarding is inherently supported by enabling routers to dynamically select the next hop for each request based on the network conditions. However, due to the limitation in network resources, a well-designed forwarding strategy is necessary to achieve satisfactory network performance. In this paper, the problem of content request forwarding in the context of NDN is naturally formulated as a semi-Markov decision problem (SMDP). Since the exact SMDP solution is intractable, a variant of reinforcement learning (RL) method integrated with function approximation by neural networks is developed to find the optimal solution for our SMDP abstract. A broad set of experimental comparisons was carried out to verify the effectiveness of the resulting forwarding strategy. The simulation results show that the proposed RL method provides an efficient solution to our SMDP-based forwarding model and our approach can further enhance network performance in comparison to existing forwarding strategies by reducing rejection ratio, network load, as well as delivery time. Load balance and differentiated services are also considered in our proposal.

Keywords: named data networking, forwarding strategy, semi-Markov decision processes, reinforcement learning

1. INTRODUCTION

With the rapid growth of Internet applications and users, current Internet architecture has exposed many defects of its initial design and shown inefficiencies in adapting to the new Internet usage. Realizing that Internet users only care about the content itself rather than where it is stored, the research communities are motivated to redesign a clean-slate network architecture that shifts from host-centric to content-centric. In this context, Named Data Networking (NDN)[1] has emerged as a most promising one, which introduces many beneficial functionalities to facilitate the dissemination of information.

As a clean-slate paradigm, NDN adopts a name-based communication model, where content is identified by an addressable name instead of host address. In the context of NDN, users trigger data delivery by issuing requests for desired contents, and the network core takes full responsibility to locate and retrieve the data. Since NDN decouples content retrieval from the location where the content is stored and makes contents in transit known to the network elements by naming data, request forwarding in NDN is conducted on a hop-by-hop basis rather than end-to-end fashion, which provides native support for decentralized control. In addition, unlike its IP counterpart, the design of NDN data plane[2] allows for multiple alternative outgoing interfaces rather than a single one with respect to each item. Thus, during the request forwarding process, routers should select the efficient interface(s) by making forwarding decisions based on the network conditions. The forwarding decisions of data plane are also called forwarding strategy, which is a key functionality for NDN to realize dynamic, adaptive and intelligent request forwarding. Since forwarding strategy plays an important role in data retrieval, how to design a strategy with low complexity while simultaneously assuring the efficiency of the forwarding process is a critical issue. In this paper, we focus on designing an intelligent forwarding strategy for NDN aiming at maximizing the network resource utilization.

☆* Corresponding author

Since the router dynamically decides the best outgoing interface for incoming requests based on the network conditions, the request forwarding optimization problem is actually a sequential decision making problem. Among the endless amount of decision making methods available in literatures, we are in favor of the Semi-Markov Decision Process (SMDP) theory[3] to develop our optimal forwarding strategy. The rationale behind this is that the network dynamic is governed by stochastic arrival and departure patterns of content requests and SMDP has proven to a successful approach to help make the best decisions in the stochastic environment. By grounding in the SMDP components, an optimal solution of SMDP generally seeks to maximize (or minimize) a certain function of rewards (or costs) over the states. Though there are two typical optimality metrics, i.e., the discounted sum of rewards and the averaged expected reward, however, in our request forwarding problem, the latter may be preferable. The intrinsic motivation is that the economic form may not be suitable for the performance measures of interest, such as the average queuing time, the average network throughput. In addition, since traditional methods for finding the optimal solution of SMDP models are often computationally intractable in practice due to the enormous size of state space, it seems to be a better alternative to solve our SMDP model for NDN request forwarding problem by the reinforcement learning (RL) approach. Moreover, in present work, artificial neural networks are adopted as the associative memory (or function approximators) to map network states to actions due to their ability to learn complex patterns, which provides a scalable implementation of the optimal policy resulting from the RL algorithm.

The key contributions of this work are as follows. First, we present an analytical framework for the request forwarding problem in the context of NDN by grounding in the theories of SMDPs, where the stochastic characterization of network requests and the beneficial features introduced by NDN are both taken into account. Second, the optimal forwarding strategy derived from our SMDP abstract, which we hereinafter refer to as SMDPF strategy, is learned through a RL approach and compressed into artificial neural networks which are embedded in NDN nodes to guide request forwarding. The resulting SMDPF strategy is content-aware and consequently supports differentiated forwarding. Finally, we verify the theoretic model by numerical analysis and simulations. The performance of the SMDPF strategy is evaluated by comparing to existing typical strategies. Experimental results illustrate that the present forwarding strategy can achieve higher network performance with regard to the given measure.

The rest of this paper is organized as follows: We first discuss the related work in Section II, and then introduce the problem statement, the necessary notations and assumptions in Section III. Section IV presented our SMDP formulation of NDN request forwarding problem and the solution structure for our SMDP abstract. Section V gives the detailed description of the reinforcement learning algorithm used to derive the optimal forwarding strategy of our SMDP model. The performance of the resulting SMDPF strategy is numerically evaluated in comparisons with several typical strategies, and simulation results are reported in Section VI. Finally, we conclude the paper and highlight our future work in Section VII.

2. RELATED WORK

NDN has received increasing attention, and there is a large body of research in this field. One of the most important topics is the design of its transmission mechanism. As reflected in the literatures, although both approaches in the control plane and the data plane can be exploited to adapt request transmission in the context of NDN, this work adheres to the data plane's forwarding functionality. The approaches in this regard that have been put forward in the literatures can be broadly classified into two categories: deterministic strategy and probabilistic strategy. In this section, we review some of the most well-known works.

As the name suggests, the deterministic strategy selects one or more optimal egress interfaces with certainty for forwarding requests according to given criterions. There are several studies belonging to this category. For instance, Yi et al.[4] sketched out an initial design of NDN adaptive forwarding strategy where interfaces are classified based on a coloring scheme, and ranked according to their response delay. Requests are always forwarded to the highest-ranked interface marked with green (or yellow). Afanasyev et al.[5]implemented several standard forwarding strategies (i.e., Broadcast, BestRoute, and NCC) in the latest version of NDN simulator, where requests are forwarded to all supplied interfaces, to the upstream interface with the lowest hop count indicated by routing mechanism, and to those interfaces with the lowest delay, respectively. One common weakness of these strategies is that the forwarding decisions are based on the routing information, but the update of routing information is costly and time-delay. Yeh et al.[6] proposed a Virtual Interest Packets (VIPs) framework for joint dynamic forwarding and caching in NDN. They applied backpressure algorithm to the virtual control plane which operates on the VIPs to guide the forwarding and caching of

Interest/Data packets in the actual data plane. However, the VIPs method increases the application complexity and the overhead it introduces is also significant. Udugama et al.[7] developed an On-demand Multi-Path Interest Forwarding (OMP-IF) strategy where multiple node-disjoint paths from the client to the content sources are identified and used simultaneously for request forwarding. The node-disjointness of paths is guaranteed by permitting each router to only use a sole interface to forward requests for each content prefix, while multipath transmission is triggered by clients via a weighted round-robin mechanism based on the delay measurement. However, the node-disjoint characteristic may not be able to make full use of network resources. Both works [8, 9] rely on reinforcement learning approach for learning to adapt to the network dynamics. They leverage probabilistic explorations to discover unaddressed cached item replicas and find the interface with lowest retrieval time in a distributed way. Then they exploit the acquired knowledge by forwarding requests to the best interface found until the moments to make forwarding decisions.

On the contrary, the probability-based forwarding strategies view the request forwarding problem as a process of allocating probabilities to all supplied interfaces and take a weighted random manner to make forwarding decisions. Literature [10] abstracted NDN forwarding process as a stochastic distributed multi-objective problem where the probability distribution over all interfaces is calculated by the ant colony optimization algorithm. The proposal performs well in reducing the content delivery time, but fails to consider the load balance of the network. The authors in [11] introduced a MDP-based forwarding scheme which uses MDP to model the forwarding process and integrates with a rate control mechanism to keep the stability of transmission queues. However, the scheme is not supported for content-aware forwarding and consequently results to performance degradation in the heterogeneous scenario which is often the case in NDN networks. In [12], an entropy-based probabilistic forwarding (EPF) strategy is put forward to make a stochastic interface selection based on multiple network metrics by introducing the concept of entropy. However, how to choose the appropriate parameter setting of the proposed algorithm for different network topologies is a question at hand before practical application. Gong et al.[13] presented a probabilistic binary tree based forwarding strategy (PBTF), which maps the forwarding process of NDN into the path selection problem of a probabilistic binary tree and employs an online machine learning method to adjust the weights of the probabilistic binary tree. Although PBTF successfully takes advantage of machine learning to improve the efficiency of request forwarding in NDN, it has limitations of trapping into a local optimum. Muralidharan et al.[14] developed an MDP-IoT model to direct the heterogeneous IoT traffic to the best interface based on the traffic type. The MDP algorithm reduces the RTT values, thereby fulfilling latency requirements of delay-intolerant applications in IoT-NDN environment. In [15], Carofiglio et al. derived a family of receiver-driven multipath congestion control strategies and of distributed request forwarding algorithms (RFA) from a multi-commodity flow problem. The basic idea of RFA is to compute the forwarding probability of a interface based on a moving average over the reciprocal count of its PIT entries with the objective of minimizing the total number of pending requests at each network node. RFA is simple, yet fails to take into account the historical state information of interfaces when making forwarding decisions, thus leading to a suboptimal solution. Posch et al.[16] designed a Stochastic Adaptive Forwarding (SAF) strategy to intelligently guide and distribute requests through network crossings circumventing link failures and bottlenecks by imitating a self-adjusting water pipe system. SAF adapts the forwarding probabilities for interfaces according to the amount of unsatisfied traffic as well as the overpressure valves to maximize the Interest satisfaction ratio.

Our work differs from these recent studies in the following aspects: first, using the theory of event-based optimization, we model the forwarding decision process in FIB of a single NDN router as an event-driven SMDP problem. Second, multi-objective optimization and differentiated forwarding are inherently supported by the design of our SMDP model, thereby the resulting forwarding strategy is flexible to cater to different application requirements. Third, we adopt a novel variant of Q-learning integrated with artificial neural networks to learn the optimal strategy based on our SMDP framework, which is effective to deal with the curse of dimensionality faced by traditional tabular reinforcement learning techniques.

3. PROBLEM STATEMENT

This section provides a brief description of the general content request processing at NDN nodes, while giving the notations and assumptions used throughout the present work.

3.1. Overview of request forwarding in named data networking

Communication in NDN is driven by the data requesters through the exchange of two types of packets: Interest packet and Data packet. To trigger data delivery, the requesters need to send out Interest packets carrying the names of the desired data, and the routers are responsible for forwarding the Interest packets towards the potential data sources in a hop-by-hop fashion. The request forwarding process is driven by NDN forwarding engines, which are composed of three data structures: the Content Store (CS), the Pending Interest Table (PIT), and the Forwarding Information Base (FIB). NDN introduces the in-network caching to facilitate content retrievals by enabling routers to temporarily cache the received data in their local CS for satisfying the same requests in the near future. To maintain stateful forwarding, routers should record the information of all the requests forwarded upstream in their PIT, which is also used for request aggregation. The FIB maps content names to the egress interface(s) available for retrieving the corresponding data. In addition, a strategy module is equipped at each NDN router, which stores all the available forwarding strategies and takes responsibility to make intelligent forwarding decisions. The basic NDN request forwarding process is performed as follows.

Upon reception of an Interest packet, the routers perform a longest name prefix match lookup on the three data structures in the order of CS, PIT and FIB. In case of a CS hit, routers will respond locally to the incoming request via Data packet and then discard the Interest packet. Otherwise, if there is a match in PIT, routers will append the arrival interface of the Interest packet to the corresponding PIT entry and discard the Interest packet due to the request aggregation function. Otherwise, if there is an exact-match FIB entry, the Interest packet will be propagated further through one or more available egress interfaces, which are determined dynamically by the predefined forwarding strategy. Accordingly, a new PIT entry will be created. Otherwise, the Interest packet will be discarded and a NACK packet will be sent back to the requester(s) notifying the cause of network problems. A router that receives a NACK packet will consult the FIB and try other alternative egress interfaces within the lifetime of this request. In addition, since Data packets are not routed but follow the reverse path of the corresponding Interest packets back to the requester(s), a router only needs to perform a lookup in its PIT when a Data packet arrives. In case of a PIT hit, it will send the received Data packet out all of the interfaces listed in the corresponding PIT entry and then remove the PIT entry. Otherwise, the Data packet will be discarded.

As can be seen from the above, forwarding decisions should be made each time a request arrives. In order to model this decision making process by using the SMDP framework, we start our analysis by introducing the necessary notations and assumptions in the following subsection.

3.2. Notations and Assumptions

Without loss of generality, we consider a NDN router with a set of interfaces $\mathcal{L} = \{1, 2, \cdots, L\}$, of which each interface l has a total capacity of B_l units of bandwidth. Assume that the content catalogue in the system is partitioned into K classes labeled as $\mathcal{K} = \{1, 2, \cdots, K\}$. Each class $k \in \mathcal{K}$ is characterized by its bandwidth requirement b_k, rejection cost h_k, and reward functions $\mathbf{C}_k = \{c_{lk}\}, \forall l \in \mathcal{L}$. The bandwidth requirement b_k may reflect either the peak transmission rate of class k requests, or their effective bandwidth. The rejection cost h_k denotes the penalty cost for rejecting a content request of class k. While the reward rate c_{lk} denotes the expected benefit earned per unit time by forwarding a request for class k content through interface l, which is not necessary a monetary one but may reflect the prioritization of different request classes or their desired quality-of-service (QoS). Due to a trade-off between optimality and scalability, we focus on designing a coarse-grained class-based forwarding strategy in this work. Furthermore, we assume that users request data according to the previously described content catalog and each content request is independent from the previous one. The router can receive content requests on any interface $l \in \mathcal{L}$ and satisfy these requests by either returning a locally stored copy of the requested data or by forwarding the requests to suitable interfaces. The request arrival process is modeled through a Markov Modulated Rate Process (MMRP)[17]. For convenience, a summary of notations used in this paper is given in Table 1 .

In what follows, we will formulate the problem of NDN request forwarding as a continuous time, average reward, finite-state semi-Markov decision problem, and consequently derive the optimal forwarding strategy that is average reward optimal.

Table 1: Notations used in this paper

Notation	Description
$\mathcal{L}$	Interfaces set
B_l	Capacity of interface l
$\mathcal{K}$	Content catalog
b_k, h_k	Bandwidth requirement and rejection cost of class k requests
c_{lk}	Reward function of forwarding class k requests through interface l
$\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{F}$	Components of SMDP: state space, action space, transition function, reward function, sojourn-time distribution function
$e_{lkz}, \mathcal{E}$	Event and event space
λ_{lkz}	Average arrival rate of event e_{lkz}
$\mathcal{L}^k$	Set of egress interfaces available for transmitting the class k requests
$\mathcal{L}_{e_{ik2}}(\mathbf{s})$	Set of available egress interfaces when event e_{ik2} occurs in state $\mathbf{s}$

4. SEMI-MARKOV DECISION PROCESS FORMULATION

For a better understanding of our proposal, we first provide a brief description of the SMDPs formulation. Generally, the SMDP works as follows: there is a state space, denoted as $\mathcal{S}$; and for each state $x \in \mathcal{S}$, $\mathcal{A}(x)$ is a finite set of control actions allowed to be taken. Suppose in state x we choose an action $a \in \mathcal{A}(x)$, then the system would transit to another state $y \in \mathcal{S}$ with probability $\mathcal{P}_{xy}(a)$. As a result, we will gain a reward or cost $\mathcal{R}(x, a)$. This process is repeated. In addition, the sojourn time at each state x, denoted by $\tau(x, a)$, follows some particular distribution function $\mathcal{F}$. The goal is to find an optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$, that maximizes (or minimizes) the expected cumulated rewards (or costs) for each state. Therefore, a SMDP is completely specified by the five tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{F}\}$. A detail background of SMDPs is introduced in [3]. In what follows, we will start to specify our SMDP framework for NDN request forwarding problem.

4.1. System State and State Space

For any time t, let $s_{lk}(t)$ be the number of forwarded unsatisfied class k requests on interface l. The state $\mathbf{s}(t)$ of the NDN router at time t is defined as the vector consisting of a list of the numbers $s_{lk}(t)$, for each l and k. Hence, the state space $\mathcal{S}$ (the set of all possible states) is defined implicitly by the requirements that each $s_{lk}(t)$ be a nonnegative integer and that

$$\sum_{k \in \mathcal{K}} s_{lk}(t) b_k \leq B_l; \forall l \in \mathcal{L}. \tag{1}$$

4.2. Decision Epochs and Expected Sojourn Time

Even though the process evolves in continuous time, we only need to consider the state of the router at the moments when certain events take place. We assume that at each time step, only one event occurs. Then we define three kinds of events of interest that occur on the data plane of NDN routers: the arrival or departure of content requests. Firstly, when the router receives a request (i.e., Interest packets or NACK packets) for the class k content from a certain interface l, we denote it by e_{lk1} if the incoming request can be served by the local CS or aggregated in the PIT; otherwise by e_{lk2}. In addition, the event corresponding to the reception of Data packets related to class k content items from the interface l is denoted by e_{lk0}, which amounts to a class k request departure. We denote by $\mathcal{E}$ the finite set of all possible events, given by formula (2).

$$\mathcal{E} = \{e_{lkz} | l \in \mathcal{L}; k \in \mathcal{K}; z \in \{0, 1, 2\}\} \tag{2}$$

Therefore, the decision epochs are those moments when there is an occurrence of the events defined above. The expected sojourn time in state $\mathbf{s}$, denoted by $\tau(\mathbf{s})$, can be given by formula (3).

$$\tau(\mathbf{s}) = \frac{\mathbf{1}}{\sum_{l,k,z} \lambda_{lkz}(\mathbf{s})} \tag{3}$$

where $\lambda_{lkz}(\mathbf{s})$ is the average arrival rate of event e_{lkz} in state $\mathbf{s}$.

4.3. *Possible Actions and Action Space*

At any decision making epoch t, given the router state $\mathbf{s}(t) \in \mathcal{S}$ and the event $e(t) \in \mathcal{E}$ occurring, a decision $a(\mathbf{s}, e) \in \mathcal{A}(\mathbf{s}, e)$ has to be made, where $\mathcal{A}(\mathbf{s}, e)$ denotes the set of possible actions in state $\mathbf{s}(t)$ when event $e(t)$ occurs. If event e corresponds to the arrival of a request, the set of possible decisions consists of the available forwarding interfaces (subject to the capacity constraints and the router state) and the rejection decision. If event e corresponds to returning data, there are no decisions to be made. Hence, the request departure epochs can be viewed as "virtual" decision epochs.

For the sake of expression, let $\mathcal{L}^k \in \mathcal{L}$ represent the set of egress interfaces supplied in FIB that are available for class k content. When the router receives a NACK packet in an event that happens in state $\mathbf{s}$, both those interfaces listed in the corresponding PIT entry and the interfaces, which failed to retrieve the corresponding data within the request's lifetime, are denoted by $\mathcal{L}^-_{e_{ik2}}(\mathbf{s})$. In other cases, we define $\mathcal{L}^-_{e_{ik2}}(\mathbf{s})=\emptyset$. In addition, we define the set of available egress interfaces when event e_{ik2} occurs given the router state $\mathbf{s}$ by $\mathcal{L}_{e_{ik2}}(\mathbf{s})$, then we have

$$\mathcal{L}_{e_{ik2}}(\mathbf{s}) = \left\{ l \middle| l \in \mathcal{L}^k - \mathcal{L}^-_{e_{ik2}}(\mathbf{s}),\ \sum_j s_{lj} b_j \le B_l - b_k \right\} \setminus \{i\}. \tag{4}$$

Before specifying the set of possible actions $\mathcal{A}(\mathbf{s}, e)$ for various circumstances, we define the actions "response with the returning data", "choose interface l to serve the incoming request", and "reject a request" by a_0, $a_l (l \in \mathcal{L})$, and a_{L+1}, respectively. Then,

1) When an event $e_{lk0} \in \mathcal{E}$ occurs regardless of the router state, the returning data will be forwarded to those interfaces listed in the matching PIT entry if any. Hence, corresponding to an event e_{lk0}, the only feasible action is a_0.

2) When the router receives a request that can be fulfilled locally, that is $e_{lk1} \in \mathcal{E}$ happens, the router would response to the request with the matching data. Therefore, action a_l is taken in these cases.

3) At the arrival of event $e_{ik2} \in \mathcal{E}$ in state $\mathbf{s}$, if $\mathcal{L}_{e_{ik2}}(\mathbf{s}) \neq \emptyset$, then the content request would be further forwarded to one of the interfaces in $\mathcal{L}_{e_{ik2}}(\mathbf{s})$, which is decided by the forwarding strategy. Otherwise, since $\mathcal{L}_{e_{ik2}}(\mathbf{s}) = \emptyset$, action a_{L+1} would be taken due to no interface available for serving the incoming request.

Therefore, we can derive the whole action space, $\mathcal{A} = \cup_{\mathbf{s}\in\mathcal{S}, e\in\mathcal{E}} \mathcal{A}(\mathbf{s}, e)$, by

$$\mathcal{A} = \{a_0, \ldots, a_l, \ldots, a_{L+1}\}. \tag{5}$$

4.4. *State Transition Probability*

The router dynamic can be completely described by the transition probabilities among the router states. For the purpose of illustration, we define two vectors of integers as follows:

$$\begin{aligned} \mathbf{s}^{\alpha\beta}_{+} &= \left(s_{11}, \cdots, s_{\alpha\beta} + 1, \cdots, s_{LK}\right) \\ \mathbf{s}^{\alpha\beta}_{-} &= \left(s_{11}, \cdots, s_{\alpha\beta} - 1, \cdots, s_{LK}\right) \end{aligned} \tag{6}$$

which represents an increase/decrease in the variable located at the position (α, β) in state $\mathbf{s}(t)$, respectively. For the Markov chain underlying the SMDP, and for any action $a \in \mathcal{A}$, there is a transition matrix $\mathcal{P}(a) = \{p(\mathbf{s}'|\mathbf{s}, a) : \mathbf{s}, \mathbf{s}' \in \mathcal{S}\}$, where $p(\mathbf{s}'|\mathbf{s}, a)$ represents the one step transition probability from state $\mathbf{s}$ to state $\mathbf{s}'$ under action a. For all feasible $\mathbf{s}, \mathbf{s}' \in \mathcal{S}$, $p(\mathbf{s}'|\mathbf{s}, a)$ is specified as follows:

$$p(\mathbf{s}'|\mathbf{s}, a) = \begin{cases} \dfrac{\lambda_{\alpha\beta 0}(\mathbf{s})}{\sum_{l,k} \lambda_{lk0}(\mathbf{s})}, & \mathbf{s}' = \mathbf{s}^{\alpha\beta}_{-}, a = a_0 \\ \dfrac{\sum_{l,l\neq\alpha} \lambda_{l\beta 2}(\mathbf{s})}{\sum_k \left(\lambda_{\alpha k1}(\mathbf{s}) + \sum_{l,l\neq\alpha} \lambda_{lk2}(\mathbf{s})\right)}, & \mathbf{s}' = \mathbf{s}^{\alpha\beta}_{+}, a = a_\alpha \\ \dfrac{\sum_k \lambda_{\alpha k1}(\mathbf{s})}{\sum_k \left(\lambda_{\alpha k1}(\mathbf{s}) + \sum_{l,l\neq\alpha} \lambda_{lk2}(\mathbf{s})\right)}, & \mathbf{s}' = \mathbf{s},\ a = a_\alpha \\ 1, & \mathbf{s}' = \mathbf{s}, a = a_{L+1} \\ 0, & otherwise \end{cases} \tag{7}$$

where $\alpha \in \mathcal{L}^\beta$.

4.5. Rewards

It is noteworthy that alternative elaborate definitions of the reward function may be possible for encouraging different desired behaviors. This is completely network dependent and must follow, first and foremost, the objectives of the designer. For the case discussed here, our main goal is to determine a rule for maximizing the utilization of router resources while efficiently balancing the offered load among the available egress interfaces.

Given the state $\mathbf{s}$ of the router, an event e, and a decision $a \in \mathcal{A}(\mathbf{s}, e)$, the router moves to a new state which will be denoted by $\mathbf{s}'$. The actual cumulative reward between the two successive decision epochs will be denoted by $r(\mathbf{s}, e, a, \mathbf{s}')$: if e corresponds to a class k request arrival and a is a decision to forward the request or rejection, then $r(\mathbf{s}, e, a, \mathbf{s}')$ can be calculated as $r(\mathbf{s}, e, a, \mathbf{s}') = \hat{\tau}(\mathbf{s}') \sum_{l,k} s'_{lk} c_{lk} - h_k \mathrm{I}_{a_{L+1}}(a)$; otherwise, $r(\mathbf{s}, e, a, \mathbf{s}') = 0$, where $\hat{\tau}(\mathbf{s}')$ is the actual sojourn time in state $\mathbf{s}'$, and $\mathrm{I}_y(x)$ is a Kronecker delta function, with

$$\mathrm{I}_y(x) = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \tag{8}$$

4.6. Performance Measurement and Optimal Solution

When a new request arrives, we need to choose a preferable interface based on the router state if multiple egress interfaces are available. The objective is to perform request forwarding in such a way that the long term average reward is maximized. Regarding the SMDP's optimal solution, it is noteworthy that it always exists and is stationary when the process has a finite state space, a finite action space, and bounded rewards as that in this paper.

We define a policy to be a mapping $\pi : \mathcal{S} \times \mathcal{E} \to \mathcal{A}$. We note that under any given policy π, the state $\mathbf{s}(t)$ evolves as a continuous time finite state Markov process. Let t_n be the time when the nth event $e(t_n)$ happens, and $\mathbf{s}(t_n)$ be the state of the router just prior to that event. (This notation is equivalent to assuming that $\mathbf{s}(t)$ is a left continuous function of time). We then define the average reward $\eta(\pi, \mathbf{s})$ of an SMDP with initial state $\mathbf{s}$ and policy π to be

$$\eta(\pi, \mathbf{s}) = \lim_{N \to \infty} \frac{\mathbb{E}_{\mathbf{s}}^{\pi} \left\{ \sum_{n=0}^{N} r\left(\mathbf{s}\left(t_n\right), e\left(t_n\right), a\left(t_n\right), \mathbf{s}\left(t_{n+1}\right)\right) \right\}}{\mathbb{E}_{\mathbf{s}}^{\pi} \left\{ \sum_{n=0}^{N} \left(t_{n+1} - t_n\right) \right\}}, \tag{9}$$

where $a\left(t_n\right) = a\left(\mathbf{s}\left(t_n\right), e\left(t_n\right)\right)$, and $\mathbb{E}_{\mathbf{s}}^{\pi}$ denotes the expectation with respect to the probability measure generated by policy π and initial state $\mathbf{s}$. Under the assumption that for all request classes the average response time is finite, the state corresponding to an empty system is recurrent under every policy. For this reason, the limit in formula (9) exists, is independent of the initial state and is equal to a deterministic constant with probability one for every stationary policy. Thus, we have $\eta(\pi, \mathbf{s}) =: \eta(\pi), \forall \mathbf{s} \in \mathcal{S}$.

A policy π^* is said to be average expected reward optimal if $\eta(\pi^*) \geq \eta(\pi)$ for every other policy π. Furthermore, we denote the average reward associated with the optimal policy π^* as η^*. In principle, the optimal policy can be obtained by solving the Bellman optimality equations for average reward SMDP problems, which take the following form:

$$\eta^* \tau(\mathbf{s}) + h^*(\mathbf{s}) = \mathbb{E}_e \left\{ \max_{a \in \mathcal{A}(\mathbf{s}, e)} \left[r(\mathbf{s}, e, a, \mathbf{s}') + h^*(\mathbf{s}') \right] \right\}. \quad \mathbf{s}, \mathbf{s}' \in \mathcal{S} \tag{10}$$

Here, $\mathbb{E}_e\{\cdot\}$ stands for the expectation with respect to the event e occurring in state $\mathbf{s}$. Furthermore, $\mathbf{s}'$ stands for the state right after the event, which is a deterministic function of $\mathbf{s}$, e, and the decision a. Under the previous assumption of a finite unichain SMDP, the Bellman equations have a unique solution and the functions $h^*(\mathbf{s}), \forall \mathbf{s} \in \mathcal{S}$ are called the optimal state-value functions. Once the optimal state-value functions $h^*(\cdot)$ are available, an optimal request forwarding policy π^* can be given by

$$\pi^*(\mathbf{s}, e) = \arg \max_{a \in \mathcal{A}(\mathbf{s}, e)} \left[r(\mathbf{s}, e, a, \mathbf{s}') + h^*(\mathbf{s}') \right]. \tag{11}$$

A lot of methods have been put forward in the literatures to obtain the optimal solution π^*. However, traditional methods, such as dynamic programming, commonly suffer from the curse of dimensionality, which means the complexity to represent (compute or store) the optimal value functions can grow intractably with the scale of the problem and consequently makes it impractical for problems with large state and action spaces. This motivates us to consider a kind of reinforcement learning methods that work with function approximations to learn the optimal value functions through iterative reinforcement.

5. SOLUTION METHODS

A successful application of SMDP relies crucially on the performance of its solution methods: it should be effective enough to converge to the optimality, but also simple to limit the convergence time. Recently, reinforcement learning has become a topic of intensive research as an alternative approach to solve large-scale problems. This method has two distinct advantages over classical methods. On the one hand, it is model-free: it can be applied to infer the optimal policy without explicitly estimating the rewards and the transition dynamics $p(\mathbf{s}'|\mathbf{s}, a)$. On the other hand, it can integrate with various function approximation methods, which can prevent memory exhaustion due to explicit storage of the values for each state-action pair.

5.1. Reinforcement Learning with Function Approximation

Formally, reinforcement learning methods use the representation of action value (also known as Q function) rather than state value. We define the optimal action-value function $Q^*(\mathbf{s}, e, a)$ as the maximum achievable expected average value by following any policy after taking action a in state $\mathbf{s}$ when event e occurs. Then the average reward Bellman equations for SMDPs can be rewritten in action-value form as formula (12).

$$Q^*(\mathbf{s}, e, a) = \mathbb{E}_e \left\{ r(\mathbf{s}, e, a, \mathbf{s}') + \max_{a' \in \mathcal{A}(\mathbf{s}', e')} Q^*(\mathbf{s}', e', a') \right\} - \eta^* \tau(\mathbf{s}), \forall \mathbf{s} \in \mathcal{S} \tag{12}$$

The basic idea behind many reinforcement learning algorithms is to estimate the action-value functions by using the above Bellman equations as an iterative update,

$$Q_{n+1}(\mathbf{s}, e, a) = \mathbb{E}_{\mathbf{s}', e'} \left[r(\mathbf{s}, e, a, \mathbf{s}') - \eta_n \tau(\mathbf{s}') + \max_{a' \in \mathcal{A}(\mathbf{s}', e')} Q_n(\mathbf{s}', e', a') \,|\mathbf{s}, e, a \right], \tag{13}$$

where η_n is the average reward, defined in formula (9), at decision epoch n, and can be estimated by taking the ratio of the total reward earned and the total simulation time till the nth decision making epoch, given by formula (14). Such value iteration algorithms converge to the optimal action-value function, $Q_n \to Q^*$, as $n \to \infty$.

$$\eta_n = \frac{\sum_{i=0}^{n} r\left(\mathbf{s}(t_i), e(t_i), a(t_i), \mathbf{s}(t_{i+1})\right)}{\sum_{i=0}^{n} \tau\left(\mathbf{s}(t_i), e(t_i), a(t_i), \mathbf{s}(t_{i+1})\right)} \tag{14}$$

In addition, an important issue in practice is how to store the action values. There are several approaches, among which the lookup table is the most straightforward method to represent action values. However, the huge memory requirement due to the large number of state-action pairs usually rules out tabular representation of the action values for a practical system. Therefore, approximate representation should be used to break the curse of dimensionality in the face of very large state spaces. Artificial neural network is naturally an attractive method due to its universal function approximation capability[18]. However, reinforcement learning is known to be unstable or even to diverge when nonlinear function approximators such as a neural network are used to represent the action-value functions[19]. Recently, Mnih et al.[20] proposed a novel method to address this instability and successfully solved the MDPs problem for a variety of game playing. In this present work, we adopted a variant of this method to learn the optimal policy of our SMDP model for the request forwarding problem in the context of NDN.

To be specific, the action value function $Q(\mathbf{s}, e, a)$ is parameterized by $Q(\mathbf{s}, e, a|\theta)$ using a neural network function approximator which is also referred as a Q-network [20], where θ are the parameters (that is, weights) of the Q-network. The Q-network is trained by adjusting the parameters θ to reduce the mean-squared error in the Bellman equations, where the optimal target values $r(\mathbf{s}, e, a, \mathbf{s}') - \eta^* \tau(\mathbf{s}') + \max_{a' \in \mathcal{A}(\mathbf{s}', e')} Q^*(\mathbf{s}', e', a')$ are substituted with the approximate target values $g = r(\mathbf{s}, e, a, \mathbf{s}') - \eta_n \hat{\tau}(\mathbf{s}') + \max_{a' \in \mathcal{A}(\mathbf{s}', e')} Q(\mathbf{s}', e', a'|\theta_n^-)$, using parameters θ_n^- from some previous iteration. Thus, the Q-learning update at iteration n uses the following loss function:

$$L_n(\theta_n) = \mathbb{E}_{\mathbf{s}, a, e, r, \mathbf{s}'} \left[(g - Q(\mathbf{s}, e, a|\theta_n))^2 \right], \tag{15}$$

in which $Q(\mathbf{s}, e, a|\theta_n)$ is the estimated action value of the neural network at iteration n, and θ_n are the parameters of the Q-network at iteration n.

So in the presence of neural networks as the function approximators, after a transition from state $\mathbf{s}$ to state $\mathbf{s}'$ (with action a), the parameters θ of the Q-network are updated by performing gradient descent on the above loss function as follows:

$$\Delta\theta_n = \alpha_n \nabla_{\theta_n} L_n(\theta_n) = \alpha_n \mathbb{E}_{\mathbf{s},a,e,r,\mathbf{s}'}\left[\left(g - Q(\mathbf{s},e,a|\theta_n)\right)\nabla_{\theta_n} Q(\mathbf{s},e,a|\theta_n)\right], \tag{16}$$

where α_n is the learning rate at the nth iteration, and ∇_{θ_n} is the vector of partial derivatives with respect to each component of θ_n. Details about the methodologies can be found in [20].

5.2. The Algorithm Structure

A complete description of the algorithm for training Q-network is given in **Algorithm1** . When an event (either a request arrival or departure) occurs, the router state $\mathbf{s}$ is identified by getting the information of all the local interfaces. Based on the state $\mathbf{s}$, a set of feasible actions is found. The information of both the router state and the event occurring is then fed into the neural networks to get the action values, based on which the router selects and executes actions according to an ε-greedy policy. That is, the router follows the greedy policy with probability $1-\varepsilon$ or selects a random action except the one with largest Q-value with probability ε. This is also referred to as exploration in the literatures[21]. When the next event occurs, the Q-network is updated, and the process is repeated.

Algorithm 1: Neural RL Algorithm for the SMDP-based Request Forwarding Problem

Initialize the replay memory
Initialize the action-value functions with random weights θ for Q and weights $\theta^- = \theta$ for $\hat{Q}$, respectively
for *episode =1,M* **do**
  Initialize the cumulative reward $CR = 0$, the total time $T = 0$, and the reward rate $\eta = 0$.
  for $n = 0,N$ **do**
    (1) if the router state at the nth time step is $\mathbf{s}_n$ and the event occurred is e_n, simulate the best action $a_n^* = \underset{a_n \in \mathcal{A}(\mathbf{s}_n,e_n)}{\arg\max}\ Q(\mathbf{s}_n,e_n,a_n|\theta_n)$ with probability $1-\varepsilon$, or randomly choose an action from the action set $\{\mathcal{A}(\mathbf{s}_n,e_n)\backslash a_n^*\}$;
    (2) Execute the chosen action and wait for the next event. Let $\mathbf{s}_{n+1}$ be the router state at the next decision epoch, τ_n be the transition time, and r_n be the cumulative reward earned as a result of taking action a_n in state $\mathbf{s}_n$ when e_n occurs;
    (3) Set router state to the new state $\mathbf{s}_{n+1}$ and $n = n+1$. Store transition $(\mathbf{s}_n, e_n, a_n, r_n, \tau_n, \eta_n, \mathbf{s}_{n+1})$ in replay memory. In case a non-exploratory action is chosen in Step (1), update the variables: $CR_{n+1} = CR_n + r_n$, $T_{n+1} = T_n + \tau_n$, and $\eta_{n+1} = CR_{n+1}/T_{n+1}$;
    (4) Sample random mini-batch of transitions $(\mathbf{s}_i, e_i, a_i, r_i, \tau_i, \eta_i, \mathbf{s}_{i+1})$ from the replay memory. Set
    $$g_i = \begin{cases} r_i, & if\ episode\ terminates\ at\ step\ i+1 \\ r_i - \eta_i\tau_i + \underset{a' \in \mathcal{A}(\mathbf{s}',e')}{\max} \hat{Q}(\mathbf{s}_{i+1},e',a'|\theta_i^-), & otherwise \end{cases}$$
    Perform a gradient descent step on $(g_i - Q(\mathbf{s}_i,e_i,a_i|\theta_i))^2$ with respect to the network parameters θ according to formula (16).
    Every C steps reset $\hat{Q} = Q$
  end
end

5.3. Complexity Analysis

In above neural RL algorithm, there are two loops, and in practice, we use a fixed maximum length of M and N for each loop, respectively. During the inner loop of the algorithm, the complexity of updating the Q-networks is determined by the number of samples, denoted by N_{sample}, and the time complexity of computing the feed-forward errors, denoted by TC_{err}. The time complexity TC_{err} is determined by the number of neurons in each layer of the neural networks. In addition, the iteration process of the inner loop will stop when a terminal state is reached even if the number of iterations is less than N. Therefore, the time complexity of training the SMDPF strategy is no greater than $O\left(M * N * N_{sample} * 2TC_{err}\right)$.

To exploit the resulting strategy, the router has to check the action-values of all available interfaces, which has computational complexity of $O(|\mathcal{A}(\mathbf{s},e)| * TC_{err})$ since each action-value is computed by the neural networks with TC_{err} complexity. Here, $|\mathcal{A}(\mathbf{s},e)|$ denotes the number of actions in the available action space $\mathcal{A}(\mathbf{s},e)$.

6. PERFORMANCE EVALUATION

In this section we investigate the performance of SMDPF strategy by comparing it to related works. Since our proposal operates at the content level, we only consider those forwarding solutions with the similar operation granularity for comparisons: the BestRoute (BR) strategy[5], the Request Forwarding Algorithm (RFA)[15], and the Stochastic Adaptive Forwarding (SAF) strategy[16]. As mentioned in Section II, these strategies are typical representatives of either the deterministic strategies or the probabilistic ones. The BR strategy follows the routing costs in the FIB and sends Interests to the lowest cost interface. Future Interests are always forwarded to same interface as long as that interface has the cheapest cost, regardless of its success in returning the requested data. The interface costs are updated by the routing protocol. The RFA strategy monitors the number of corresponding pending Interests of each name-prefix per interface, and uses the moving average over the reciprocal count of the PIT entries as the forwarding probability of each interface. The SAF employs overpressure valves to adapt the forwarding probability of each interface, and uses a feedback mechanism to decrease the traffic over congested links.

The purpose of our performance evaluation is twofold: (i) provide a better knowledge of the functionalities of our SMDP framework with a series of numerical experiments conducted by Matlab, and (ii) evaluate the performance of the proposed SMDPF strategy through comparing to above comparators in different respects in a realistic scenario simulated in ndnSIM[22], an ns-3 simulator module which implements the NDN protocol stack.

6.1. Numerical Experiments

Scenario 1: Performance under Various Load Levels

In the first experiment set, we perform the performance evaluations using Matlab by varying the offered load level. We consider a NDN router with four interfaces which have the fixed capacity of 60, 120, 80, and 130 bandwidth units respectively. In addition, we assume that there are three request classes involved in the system and requests of any classes are allowed to be forwarded to all of the four interfaces. The router receives requests from all interfaces and makes forwarding decisions according to the predefined forwarding strategy. For each tested strategy, various scenarios with different offered load level were considered respectively. The parameter settings used in this case study are given in Table 2 . The reward functions shown in Table 2 depict that the reward generated by a request is proportional to its average response rate, which amounts to that the performance measure considered in this scenario is the average total throughput. Moreover, the same increasing rates of reward functions c_{lk} and the rejection cost h_k for all request classes represent a scenario where the differentiation of request classes is not considered. This experiment set is designed to evaluate the ability of the strategies to find the optimal egress interfaces and balance the requests among all interfaces.

Table 2: Parameters for Scenario 1

Request Class k	Bandwidth Demand b_k	Rejection Cost h_k	Reward Function c_{lk}				Average Response Rate μ_{lk}			
			Face#1	Face#2	Face#3	Face#4	Face#1	Face#2	Face#3	Face#4
1	1.00	10.00	2.00	4.50	9.00	7.50	0.020	0.045	0.090	0.075
2	2.00	10.00	6.00	8.50	2.50	4.00	0.060	0.085	0.025	0.040
3	5.00	10.00	3.50	7.00	10.50	5.00	0.035	0.070	0.105	0.050

As for the exact architecture of the Q-networks used in our experiments, the input layer consists of fifteen input neurons, among which twelve input neurons corresponds to the state variables while the other three corresponds to the identification of event. The hidden layer is fully-connected and configured with 10 neurons with sigmoid activation property. The number of hidden layer neurons is obtained after considerable experiments. The output layer is a fully-connected linear layer with a single output for each valid action. Unless otherwise specified, the weights of the neural networks are initialized to random numbers in the interval [-0.1, 0.1], and an ε-greedy policy with ε=0.05 is adopted in our experiments. After training, the weights of the neural networks were fixed, and the experiment was rerun using the trained neural networks to choose actions without any exploration or learning. The results were averaged over multiple runs.

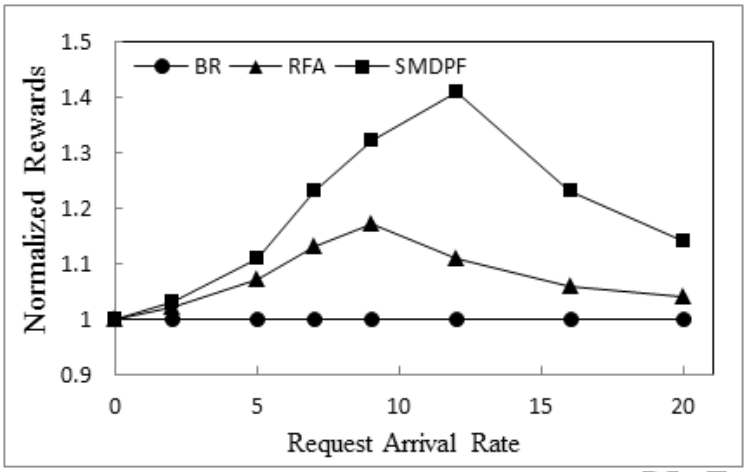


Figure 1: Normalized average reward

Figure 1 reports a representation of the average rewards, defined by formula (9), of the tested strategies normalized by that of BR strategy, as we vary the total request arrival rate λ_{total} from 0 to 20. We can note, from Figure 1 , that the proposed SMDPF strategy leads to better results than the other two tested strategies regardless of the request arrival rate. The abilities of our proposal that contribute to this performance gain are that it is not only able to take the states of all interfaces as a whole into account when making request forwarding decisions, but also able to achieve a faster forwarding adaptation to network dynamics, thereby resulting in more efficient utilization of the network resources. While in the case of a low request arrival rate, the performances of these approaches are almost the same. This is attributed to the fact that at light offered load, the requests are too rare to contribute to the performance. In addition, it can be also seen that before the router is saturated, the performance gains of SMDPF over the other two strategies become more and more significant as the total request arrival rate increases. This is because that the heavier the offered load, the more requests are forwarded to the optimal interface in such cases. However, the gain is less significant when the offered load is high enough to saturate the router. This is understandable because for a saturated router there is no extra resources for the subsequent requests, leading to an increase in the blocking probability and a consequent decline in the performance gain.

In Table 3 , we present the experimental results of the tested strategies given the total request arrival rate λ_{total} = 9.55, which amounts to a medium load level. The load factor is defined as the ratio of the average load over a given period to the total bandwidth capacity of the router. The rejection ratio denotes the estimated average ratio between the number of rejected requests and that of received requests over a given period. The average delivery time per content represents the average time elapsing between the expression of a content request and the reception of the corresponding data. The interface average throughput is defined as the time average data rates on individual interfaces. From Table 3 , we can make the observation that both SMDPF and RFA outperform BR in all respects. Furthermore, SMDPF achieves the best performance among these strategies. Specifically, it provides a performance improvement of 11.6% with respect to BR strategy, and 7.7% with respect to the RFA strategy in terms of the load factor. This can be explained by the Interface Average Throughput column in Table 3 , where we can see that SMDPF can find the best interface and make use of multiple egress interfaces to balance the requests received by the router, while BR does not identify the optimal path and overuses the suboptimal one. In addition, since RFA splits the requests over the heterogeneous interfaces so it also achieves a lower average load factor than BR strategy by exploiting alternative interfaces. As a result, the rejection ratio of BR strategy is higher than that of the other two strategies, as shown in Table 3 . Moreover, due to the statistical properties of the RL algorithm, SMDPF strategy can not only preserve the advantages of probabilistic strategy, namely exploring potential better path via rich connectivity, but also distribute the requests over all interfaces in a more reasonable fashion, thus realizing higher performance than RFA strategy. In addition, since the reward functions considered here is a good indication of the interface's service rate, therefore we also observe SMDPF outperforms other algorithms by improving the delivery time with 37.2% in comparison to BR

strategy and 20.7% in comparison to the RFA strategy, as shown in Table 3 .

Table 3: Performance Comparisons Given the Total Request Arrival Rate λ_{total} = 9.55

Forwarding Strategy	Average Reward	Load Factor	Rejection Ratio	Average Delivery Time /ms	Interface Average Throughput			
					Face#1	Face#2	Face#3	Face#4
BR	806.15	0.95	0.13	21.93	1.89	7.47	3.75	5.69
RFA	910.38	0.91	0.08	17.36	2.27	7.61	4.67	6.08
SMDPF	1045.26	0.84	0.02	13.77	2.63	8.23	7.24	5.01

Scenario 2: Differentiated Request Forwarding

Clearly, differentiated services of request forwarding need to be considered in the context of NDN. That is, for a well-designed forwarding strategy, it should be adaptive to give priority to the requests with higher reward. To quantify the potential of the proposed SMDPF strategy, we carried out a second experiment set on the same router model used in the previous scenario. However, in this case, we consider the request classes with different characteristics, distinguished by their reward functions. The parameters of different request classes are given in Table 4 , while other parameters remain unchanged as in the previous scenario. Note that the requests of class 3 are much more valuable than the other two classes. This case study is characterized by a high offered load and by requests of one class having a much higher reward than requests of the other classes. In Table 5 , we compared the experiment results of the tested strategies in terms of the average reward, rejection ratio, and average delivery time.

Table 4: Parameters for Scenario 2

Request Class k	Bandwidth Demand b_k	Rejection Cost h_k	Reward Function c_{lk}				Average Response Rate μ_{lk}			
			Face#1	Face#2	Face#3	Face#4	Face#1	Face#2	Face#3	Face#4
1	1.00	1.00	0.25	0.35	0.40	0.50	0.025	0.035	0.040	0.050
2	2.00	4.00	1.60	1.80	1.20	0.40	0.040	0.045	0.030	0.010
3	5.00	25.00	16.25	17.50	26.25	21.25	0.065	0.070	0.105	0.085

Table 5: Performance Comparisons with Heterogeneous Requests

Forwarding Strategy	Average Reward	Rejection Ratio	Average Delivery Time /ms	Class Rejection Ratio		
				Class#1	Class#2	Class#3
BR	353.57	0.26	32.68	0.06	0.21	0.63
RFA	427.65	0.19	29.04	0.05	0.18	0.46
SMDPF	574.99	0.11	24.23	0.05	0.22	0.16

As can be seen from Table 5 , SMDPF achieves the highest average reward again. This can be partially explained by the fact that SMDPF reduces the total rejection ratio significantly, which is about 57.7% lower than BR, and 42.2% lower than RFA, as specified in the third column of Table 5 . As expected, we observe that SMDPF has an advantage over the other two strategies with a performance gain of 25.9% and 16.6%, respectively, in terms of the delivery time. However, compared with *Scenario 1*, there is a little performance degradation in terms of reducing the average delivery time. This is attributed to the fact that the reward functions used in this simulation set are not only determined by the service rate, but also by the request type. Moreover, by inspecting the rejection ratios for individual request classes, we note that, as expected, the request class with the highest reward saw a largest reduction in the average rejection ratio achieved by SMDPF. However, the other two strategies failed to identify the most valuable request class, and thus resulted in much lower average rewards. This verified the effectiveness of the proposed algorithm in terms of

providing differentiated services for request forwarding, and also convinced us of the importance of a differentiated forwarding for NDN networks.

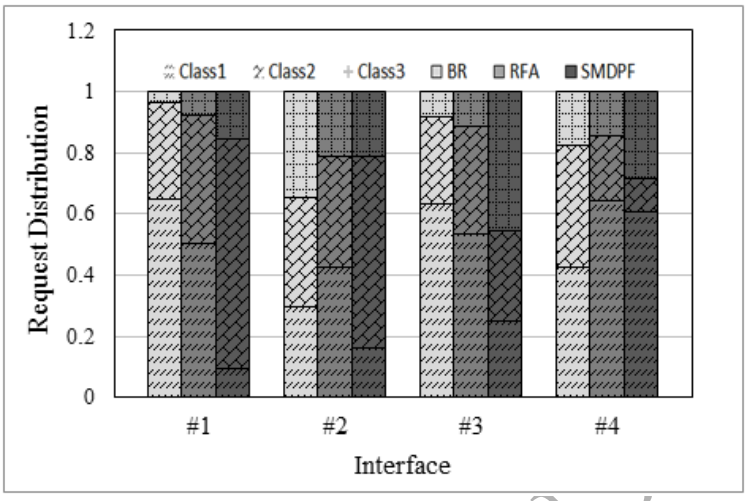


Figure 2: Request distribution of the tested strategies on individual interfaces

Figure 2 confirms the statement aforementioned, in which the request distribution over interfaces under the three forwarding strategies is illustrated. By comparing the percentages of requests served on each interface achieved by SMDPF strategy, we can observe that for every request class, a large percentage of the requests are forwarded to their preferable interface. This indicates that SMDPF indeed performs a differentiated forwarding. In addition, the differences between SMDPF and the other two strategies in terms of the amount of satisfied requests of each class show that SMDPF prefers requests with higher reward. This happens because that SMDPF strategy takes into account the states of all interfaces, and therefore, it can dynamically reserve resources for high value requests according to different request patterns.

6.2. Realistic Simulations

To quantify the benefits of the proposed strategy, we evaluate the mentioned strategies in a realistic scenario using ndnSIM. We generate a random network topology, as shown in Figure 3 , consisting of 50 nodes using the Erdõs Rényi model[23] to run the simulations. The probability of creating a link between two nodes was set to 0.08. For each link, a random bandwidth between $2Mbps$ and $5Mbps$ is assigned and the link latency is randomly drawn from a uniform distribution limited by the interval $[5ms, 20ms]$. In addition, we randomly place 40 clients and 4 servers in the network. Clients are configured to request contents with a rate ranging from 15 to 45 (requests per second), and the popularity of the requested contents follows the Zipf distribution with $\alpha = 0.7$. Clients randomly start to request contents within the first 20 seconds of the simulation. Servers are configured with a data repository that stores a non-overlay subset of the content catalog in the system permanently. Routers in the network are equipped with a finite cache size and cache every packet they receive. For what concerns the caching policies, we use the Least Recently Used (LRU) cache replacement policy[24]. The characteristic parameters of different request classes are predefined at all routers. The necessary statistical information for SMDPF is provided by a local controller, which monitors the requests received and satisfied by interfaces. The Q-Networks indicating the forwarding probabilities of each interface for individual content requests are updated based on sample paths periodically. In addition, for SAF strategy, we use the exemplary throughput-based measure demonstrated in [16], which maximizes the throughput by investigating the Interest satisfaction ratio on individual interfaces. All other simulation parameters have remained their default values. Each simulation scenario lasts 300 seconds and all performance metrics are obtained through the mean of the 20 runs by ignoring the first 20 seconds in each run to eliminate any transient effect during the warm-up phase.

Figures 4 , 5 , 6 , 7 and 8 depict the results under different levels of background traffic load and different settings of cache size in terms of the average reward per node, the average Interest satisfaction ratio per node, the average hop count per satisfied Interest, the average delivery time per content and the load balancing factor, respectively.

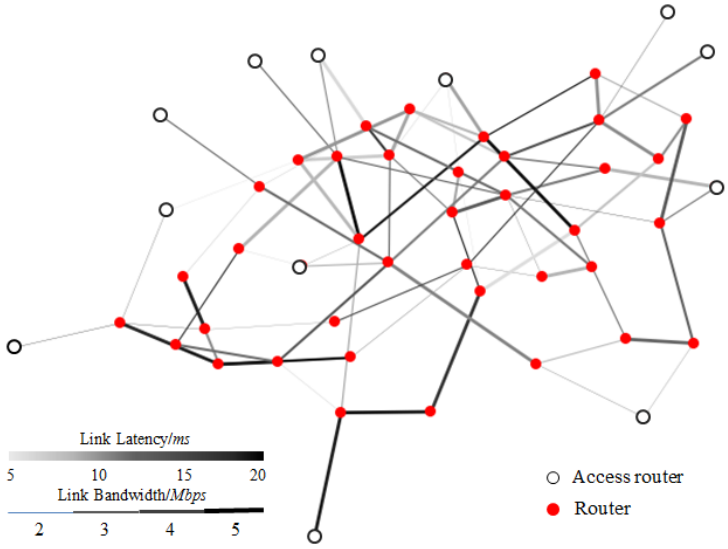
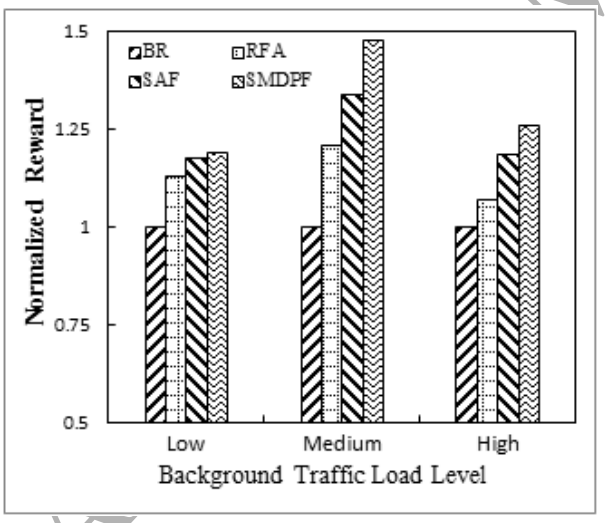
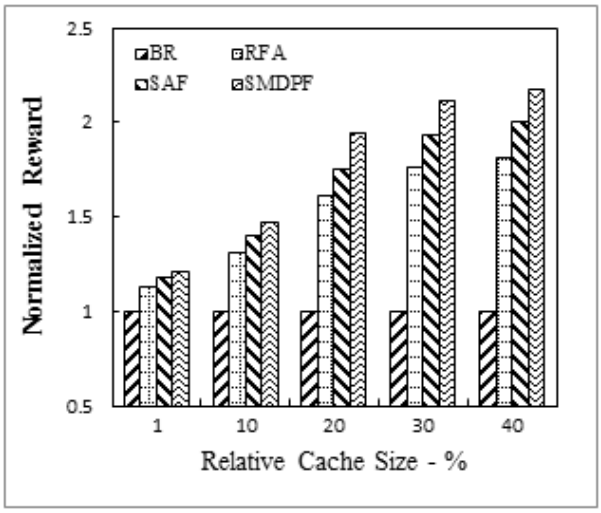


Figure 3: Realistic topology

The network reward adopts the same definition as in previous subsection. The Interest satisfaction ratio denotes the ratio between received Data packets and generated Interests by all clients. The hop count provides the number of links a Data packet traverses to satisfy an Interest issued by a client considering cache hits. The delivery time represents the average time to retrieve a desired data. The load balancing factor is defined as the coefficient of variation, $CoV(f) = stdev[f(v)]/\mathbb{E}[f(v)]$, where $f(v)$ denotes the number of requests forwarded at node v.



(a) Cache size 10% of the catalog.



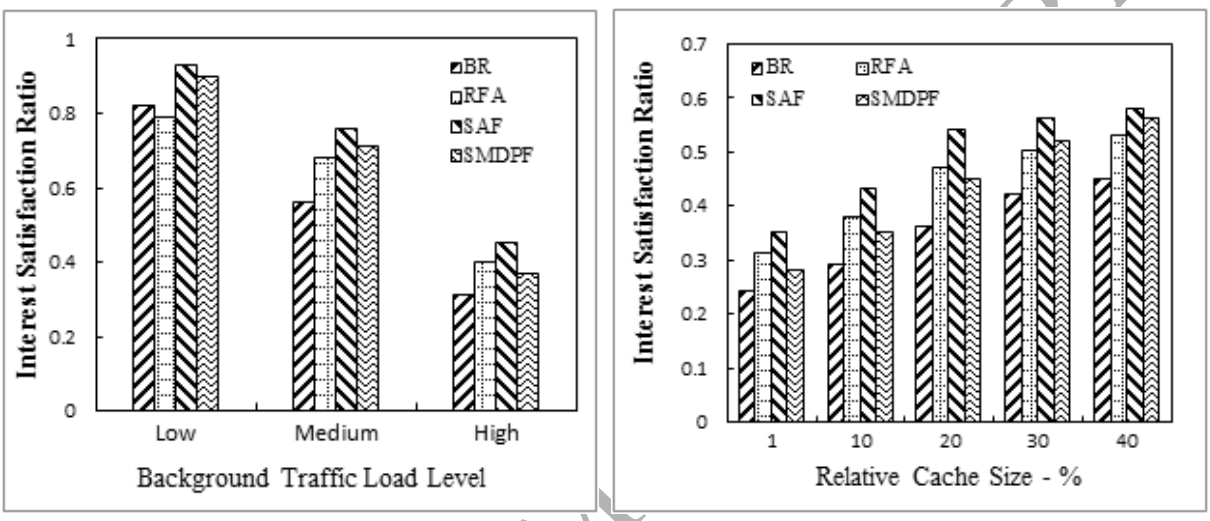
(b) High traffic load level.

Figure 4: Average reward (higher is better)

It is evident in Figure 4(a) that SMDPF outperforms the other strategies in all simulation scenarios in terms of network reward measure regardless of the level of traffic load. As expected, the three tested probabilistic strategies achieve higher network rewards than BR strategy due to a more reasonable use of network resources by adaptively selecting the potential better paths based on the network conditions for specific considerations. In addition, although

both RFA and SAF also support adaptive forwarding, SMDPF's good performance against them can be explained by its ability to offer differentiated services.

Figure 4(b) shows the influence of the cache size on the network reward for each strategy. Our SMDPF shows better results than the other schemes, and the performance differences between the tested strategies get more and more significant with the increase of cache size. In addition, the network reward sharply increases as the cache size increases from 1% to 20%. This is because the caching environment enriches the diversity of temporary content replicas, which also testifies that our proposal can better exploit the potential data sources. However, the influence of in-network caching becomes smaller and smaller when the cache size keeps on increasing due to the limitations of link bandwidth resources and the request patterns of clients.



(a) Cache size 10% of the catalog.

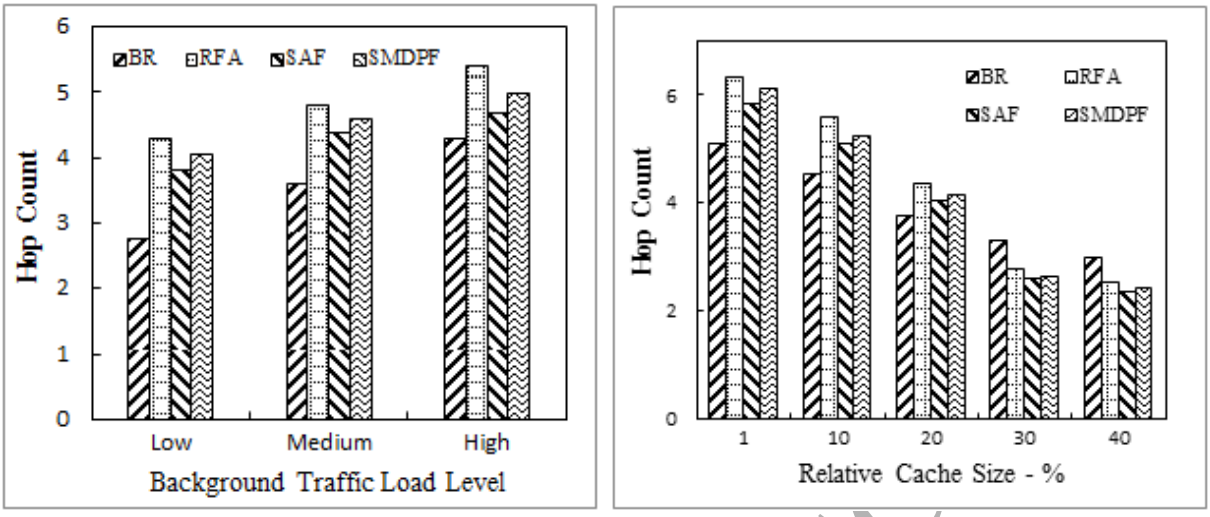
(b) High traffic load level.

Figure 5: Average Interest satisfaction ratio per node (higher is better)

Figure 5(a) illustrates the results of average Interest satisfaction ratio under different background traffic load levels. One can observe that SAF maintains the highest average Interest satisfaction ratio regardless of the network resource conditions, since it is designed to maximize the throughput at every node in the network. SMDPF achieves a little lower satisfaction ratio than SAF especially in the cases of resource shortages due to the differentiated services it imposes on various request classes. However, considering the performance gain regarding the network reward, this marginal difference is acceptable, especially when the profits of different request classes vary greatly. Interestingly, RFA sees a slight performance improvement with the increase of background traffic load. This may be attributed to the fact that RFA does not store any long term information of interfaces, leading to less efficient in scenarios with higher network resources. In addition, the poorest behavior of BR is due to link congestions caused by being overused since BR sticks to the single path with the lowest hop count from clients to the servers until exhausting the link resources.

Figure 5(b) reports the average Interest satisfaction ratio as a function of the cache size. For a configuration with small cache size, similar results are observed as those for the high traffic load level case shown in Figure 5(a) . Yet, it is noteworthy that the Interest satisfaction ratio of our SMDPF strategy increases gradually and eventually outperforms the RFA algorithm when we vary the cache size from 1% to 40%. This can be explained by the fact that in-network caching enables a more efficient utilization of the rich connectivity supported by NDN protocol. In addition, although an increase of the cache size indeed leads to better performances, however, the performance improvement with regard to this metric provided by in-network caching is bounded. This is attributed to the volatility of the cached copies and the ability of forwarding strategies in adapting to network dynamics.

As seen from Figure 6(a) , BR strategy beats the other strategies by providing the lowest average hop count. This is attributed to the fact that the capacity setting of in-network caching is relatively small in our simulations, therefore the performance gain provided by the in-network caching environments in terms of this metric is limited.



(a) Cache size 10% of the catalog.

(b) High traffic load level.

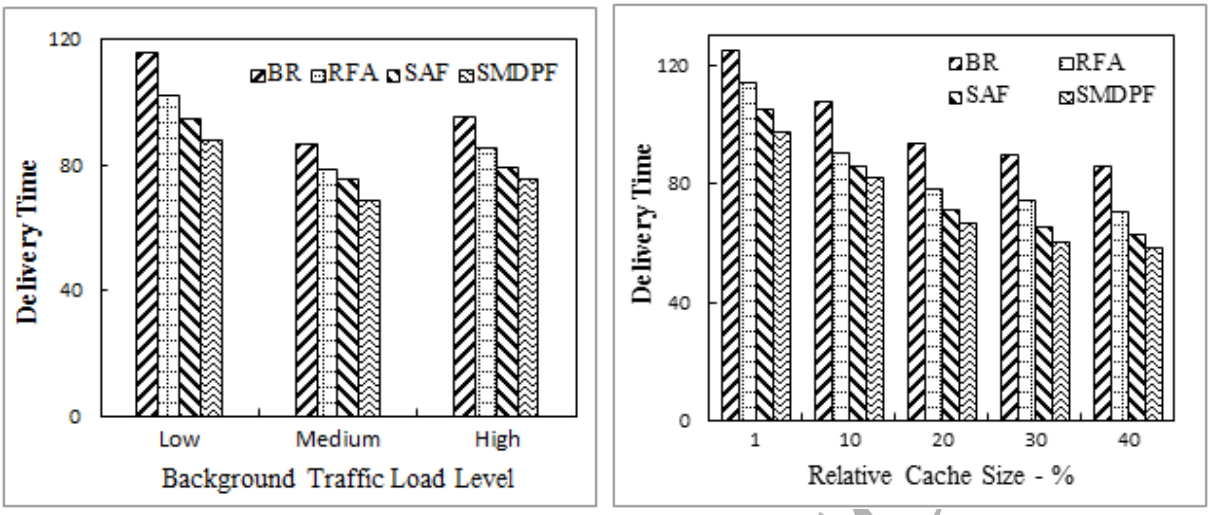
Figure 6: Average hop count per satisfied Interest (lower is better)

In addition, the fact that probabilistic strategies make use of alternative paths, which are not the shortest ones, for achieving the optimal behaviors regarding specific metrics of interest, may partially explain this phenomenon. The reason for the SFA's performance gain regarding this metric against RFA is that SAF focuses on paths that provide better performance rather than distributes the load equally on all available paths like RAF, leading to the reduction of a fraction of unnecessary detours. This is also reflected by Figure 8 depicting the load balancing behaviors of the tested strategies. In addition, compared to SAF, the higher hop count for SMDPF can be partially explained by the additional detours that SMDPF imposes on low priority requests with the aim to reserve resources for requests with high priority in order to maximize network rewards. Finally, it is obvious that BR strategy sees a fastest performance degradation with the increase of traffic load level in terms of hop count compared with other tested strategies. This can be illustrated by the fact that BR strategy makes less efficient use of the in-network caching than other schemes as the traffic load increases.

Figure 6(b) shows the trend of hop count as we vary the cache size from 1% to 40%. It is intuitive that in-network caching would shorten the distance between clients and content replicas, thus reducing the average hop count. The increase in cached replica availability, due to a larger cache size, makes the tested probabilistic strategies eventually outperform BR strategy. In addition, the differences in average hop count among the tested probabilistic strategies are narrowed by in-network caching environment as well. However, due to content redundancy in the caching environment, in-network caching almost no longer increases replica availability when the cache size ranges from 30% to 40%. Therefore, the performance improvement in terms of hop count resulted from increasing the cache size is small in these cases.

Figure 7(a) reports the average delivery time as a function of the network load. Overall, the probabilistic strategies perform better than deterministic strategies due to the capability of taking advantage of the rich connectivity provided by NDN protocol. Again, SMDPF shows the best results in all cases with different traffic load levels, and SAF is the strongest competitor to SMDPF with respect to this metric. In addition, as we can see from the plot, the trend of average delivery time for all tested strategies first declines and then grows when the traffic load level varies from low to high. This is resulted from the fact that the network connectivity and in-network caching play an important part in reducing the content delivery time in the case of medium traffic load. However, when the traffic load grows continually, network congestion is gradually built up and consequently results in the growth of delivery time, in which case the delivery time is mainly determined by the queuing delay.

In Figure 7(b) , we study the impact of cache size on the delivery time. We observe that SMDPF outperforms the other strategies with respect to this metric in all cases like those shown in Figure 7(a) . The delivery time sharply

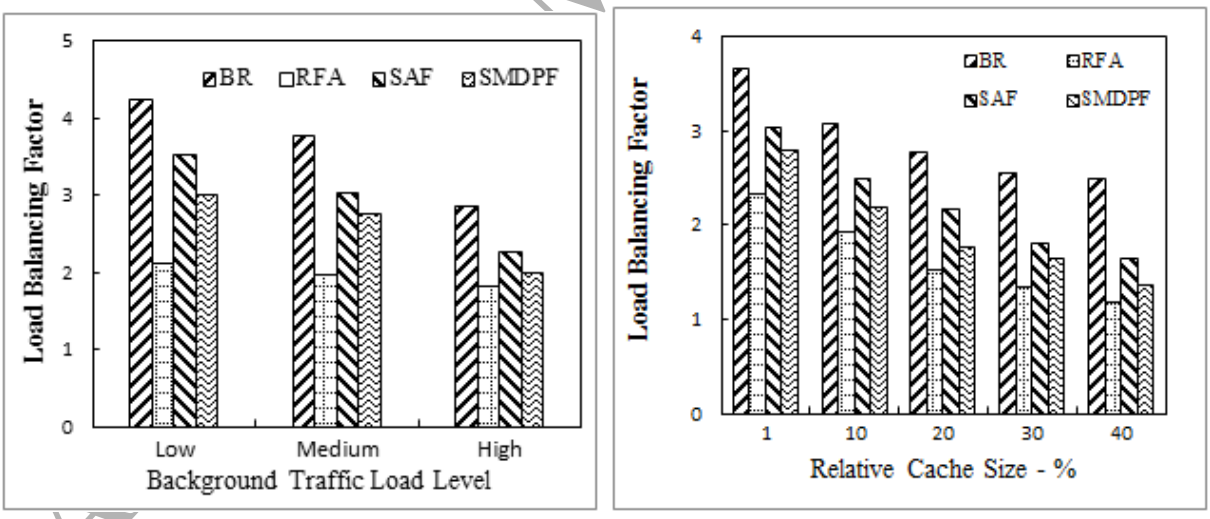


(a) Cache size 10% of the catalog.

(b) High traffic load level.

Figure 7: Average delivery time per content (lower is better)

decreases as the cache size increases until reaching a certain point. After which, additional cache capacity does not provide any more benefits. A higher cache hit ratio achieved by the tested forwarding strategies, which is reflected in Figure 6(b) , contributes to the reduction of delivery time. This also proves that in-network caching can play an important role in alleviating the network congestion, yet requiring intelligent cache management policies rather than increasing the cache size naively.



(a) Cache size 10% of the catalog.

(b) High traffic load level.

Figure 8: Load balancing factor (lower is better)

Figure 8(a) depicts the load balancing behavior of the tested forwarding strategies. It is clear that since BF only focuses on the shortest paths from the clients to the content providers, and consequently leads to the worst load balancing performance. While RFA shows an advantage over the other strategies in balancing the network load. This can be explained by its weighted random forwarding mechanism based on the number of pending requests, which

distributes requests on all routes supplied by the FIB in a relatively equal manner. In addition, SAF is also able to provide effective load balancing. However, in contrast to RFA, SAF tries to identify and exploit the better performance routes as much as possible according to the request satisfaction ratio. SMDPF achieves a kind of load balancing similar to SAF with a slight performance gain due to its content-aware request forwarding. Moreover, from Figure 8(b) , we observe a better load balancing behavior for all strategies when we increase the cache size. This is attributed to the fact that the increase in cache hit ratio reduces the load on the network and results in a relatively equal distribution of network load.

7. CONCLUSION

In this paper, we have proposed an optimization model based on the SMDP framework for the problem of content request forwarding in the context of NDN networks. By devising proper reward functions, our SMDP model is able to achieve different ambitions of the designers, such as load balance and differentiated services. The deficiencies of the classical methods in handling large state spaces motivate us to choose a model-free reinforcement learning solution to solve this problem. In addition, artificial neural networks are used in the present work to facilitate implementation of the algorithm. Simulations of various case studies were carried out to illustrate the effectiveness of the proposed scheme. Further study is in progress to study the performance of the proposed algorithm in more extensive aspects, for instance taking into account network topology, caching policy, and content popularity distribution, etc. It is also very interesting to consider other optimization algorithms for SMDP solutions.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of P.R. China under grant Nos.61233003, 61673360, and in part by the Equipment Pre-Research Foundation of China under grant No.61403120201.

References

References

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang. Named data networking. ACM SIGCOMM Computer Communication Review, 2014, 44(3), 66-73.
[2] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang. A Case for Stateful Forwarding Plane, Computer Communications, 2013, 36(7), 779-791.
[3] Baykal-Grsoy M. Semi-Markov Decision Processes [J]. Wiley Encyclopedia of Operations Research and Management Science, 2010.
[4] Yi C, Afanasyev A, Wang L, et al. Adaptive forwarding in named data networking [J]. ACM SIGCOMM computer communication review, 2012, 42(3), 62-67.
[5] Afanasyev A, Shi J, Zhang B, et al. NFD developers guide [J]. Dept. Comput. Sci., Univ. California, Los Angeles, Los Angeles, CA, USA, Tech. Rep. NDN-0021, 2014.
[6] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, D. Leong. VIP: a framework for joint dynamic forwarding and caching in named data networks, Proceedings of 1st international conference on information-centric networking, ACM, 2014, 117-126.
[7] A. Udugama, X. Zhang, K. Kuladinithi, and C. Goerg. An On-demand Multi-Path Interest Forwarding Strategy for Content Retrievals in CCN, in Network Operations and Management Symposium (NOMS), 2014 IEEE, 1-6.
[8] R. Chiocchetti, D. Perino, and G. Carofiglio. INFORM: A dynamic interest forwarding mechanism for information centric networking, Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking (ICN13), Hong Kong, Aug. 2013.
[9] Ian Vilar, Igor Monteiro Moraes. A forwarding strategy based on reinforcement learning for Content-Centric Networking [C]. 2016 7th International Conference on the Network of the Future (NOF), November 2016, DOI10.1109/NOF.2016.7810121
[10] Chengming L I, Wenjing L I U, OKAMURA K. A greedy ant colony forwarding algorithm for named data networking [J]. Proceedings of the Asia-Pacific advanced network, 2013, 34, 17-26.
[11] Su J, Tan X, Zhao Z, et al. MDP-based forwarding in Named Data Networking [C]//Control Conference (CCC), 2016 35th Chinese. IEEE, 2016, 2459-2464.
[12] Kai Lei, Jiawei Wang, and Jie Yuan. An entropy-based probabilistic forwarding strategy in named data networking. In Communications (ICC), 2015 IEEE International Conference on, 2015.
[13] Gong L, Wang J, Zhang X, et al. Intelligent Forwarding Strategy Based on Online Machine Learning in Named Data Networking [C]//Trustcom/BigDataSE/I SPA, 2016 IEEE. IEEE, 2016, 1288-1294.
[14] Muralidharan S, Roy A, Saxena N. MDP-IoT: MDP based interest forwarding for heterogeneous traffic in IoT-NDN environment [J]. Future Generation Computer Systems, 2018, 79, 892-908.
[15] Carofiglio G, Gallo M, Muscariello L. Optimal multipath congestion control and request forwarding in information-centric networks: Protocol design and experimentation [J]. Computer Networks, 2016, 110, 104-117.

[16] Posch D, Rainer B, Hellwagner H. SAF: Stochastic Adaptive Forwarding in Named Data Networking [J]. IEEE/ACM Transactions on Networking, 2016.
[17] G. Carofiglio, M. Gallo, L. Muscariello, D. Perino. Modeling data transfer in content-centric networking, Proceedings of the 23rd International Teletraffic Congress, ITC11, 2011.
[18] Kůrková V. Kolmogorov's theorem and multilayer neural networks [J]. Neural networks, 1992, 5(3), 501-506.
[19] Tsitsiklis, J. Roy, B. V. An analysis of temporal-difference learning with function approximation. IEEE Trans. Automat. Contr. 1997, 42, 674-690,
[20] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518(7540), 529-533.
[21] Bertsekas D P, Tsitsiklis J N. Neuro-Dynamic Programming. Athena Scientific, Belmont, MA, 1996.
[22] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang. ndnSIM 2.0: A new Version of the NDN Simulator for NS-3, Tech. Report NDN-0028, 2015, http://named-data.net/publications/techreports/.
[23] P. Erdõs and A. Rényi. On Random Graphs I, Publicationes Mathematicae, 1959, 6, 290-97.
[24] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie. Optimal Cache Allocation for CCN, Proceeding of 21st IEEE ICNP, 2013.

Jinfa Yao is currently a doctoral student in Control Engineering at School of Information Science and Technology, University of Science and Technology of China, Hefei, China. His research interests include network modeling and performance optimization.

Baoqun Yin received his B.S. degree in fundamental mathematics from Sichuan University, Chengdu, China, in June 1985, his M.S. degree in applied mathematics from University of Science and Technology of China, Hefei, China, in May 1993, and his Ph.D. degree in pattern recognition and intelligent system from University of Science and Technology of China, Hefei, China, in Dec. 1998.He is currently a professor of Control Science and Engineering at Department of Automation, University of Science and Technology of China, Hefei, China. His current research interests include discrete event dynamic systems, Markov decision processes, queuing systems, and network resource management and optimization.

Xiaobin Tan is currently an associate professor of Control Science and Engineering at Department of Automation, University of Science and Technology of China, Hefei, China. His current research interests include Information security, wireless multi-hop networks and future network architectures.