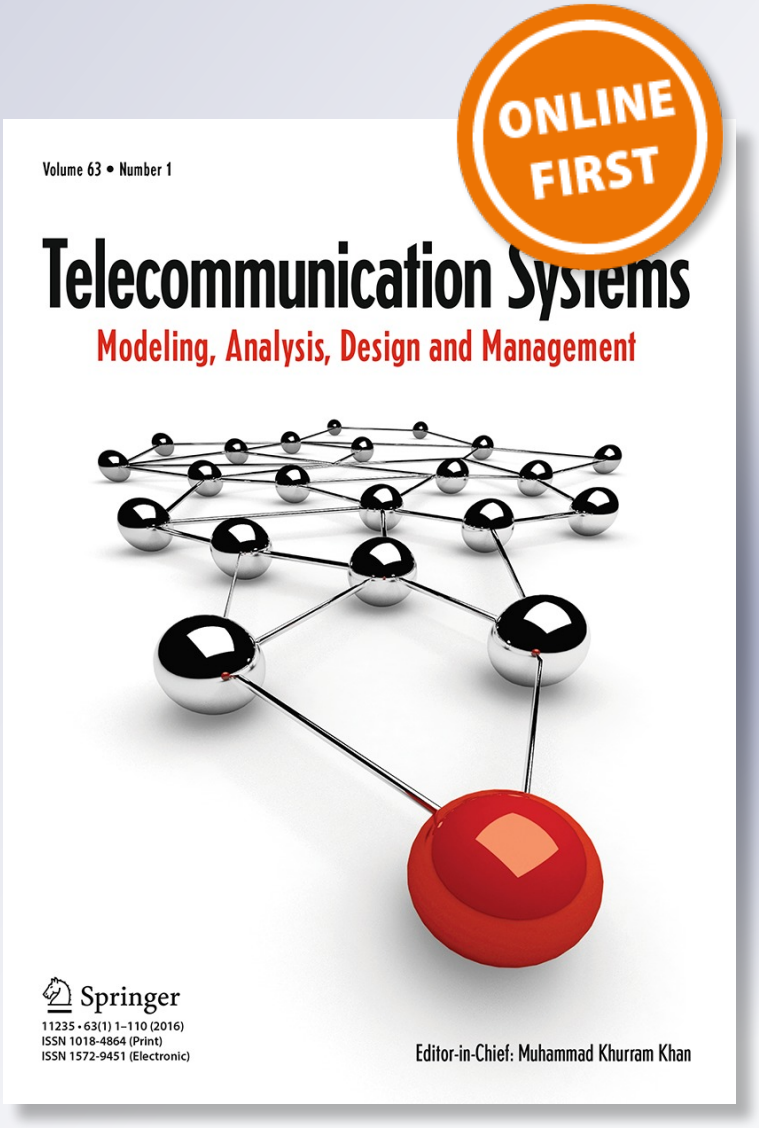
A learning automata and clustering-based routing protocol for named data networking

Zeinab Shariat, Ali Movaghar & Mehdi Hoseinzadeh

Telecommunication Systems
Modelling, Analysis, Design and Management

ISSN 1018-4864

Telecommun Syst
DOI 10.1007/s11235-016-0209-8



Springer

Telecommun Syst
DOI 10.1007/s11235-016-0209-8

A learning automata and clustering-based routing protocol for named data networking

Zeinab Shariat[1] · **Ali Movaghar**[2] · **Mehdi Hoseinzadeh**[1]

Abstract Named data networking (NDN) is a new information-centric networking architecture in which data or content is identified by a unique name and saved pieces of the content are used in the cache of routers. Certainly, routing is one of the major challenges in these networks. In NDN, to achieve the required data for users, interest messages containing the names of data are sent. Because the source and destination addresses are not included in this package, routers forward them using the names that carried in packages. This forward will continue until the interest package is served. In this paper, we propose a routing algorithm for NDN. The purpose of this protocol is to choose a path with the minimum cost in order to enhance the quality of internet services. This is done using learning automata with multi-level clustering and the cache is placed in each cluster head. Since the purpose of this paper is to provide a routing protocol and one of the main rules of routing protocol in NDN is that alternative paths should be found in each path request, so, we use multicast trees to observe this rule. One way of making multicast trees is by using algorithms of the Steiner tree construction in the graph. According to the proposed algorithm, the content requester and content owners are the Steiner tree root and terminal nodes, respectively. Dijkstra's algorithm is one of the proper algorithms in routing which is used for automata convergence. The proposed algorithm has been simulated in NS2 environment and proved by mathematical rules. Experimental results show the excellence of the proposed method over the one of the most common routing protocols in terms of the throughput, control message overhead, packet delivery ratio and end-to-end delay.

Keywords Named data networking · Content centric networking · Learning automata · Multi level clustering · Steiner tree · Routing protocol

1 Introduction

With the growth of digital media and its growing influence in the internet world, new challenges have emerged in the field of internet commerce. Customers expect to receive the content like images and videos which have high traffic on the network at the least time. With the spread of the internet, web services often suffer from congestion and bottlenecks, and intractable levels of traffic flow may increase and a lot of requests get lost in the process.

Host-to-host internet protocol (IP) model is not enough to support content distribution, peer-to-peer applications, real-time media and social networks. Internet use has changed from a low messaging and information sharing system to a high content distribution system. Hence, a large number of users request very large amounts of identical and overlapping information. So it seems that the implementation of a content distribution network on a host-to-host network is very inefficient. This is because each piece of content traverses a complete distribution chain from the producer to the consumer [1]. Moreover, some problems are encountered in addressing and routing IP addresses and are summarized as follows [1,2]:

(1) Creating heavy traffic and an explosion in duplicate requests and popular content.

✉ Zeinab Shariat
zshariat@alum.sharif.edu

[1] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

[2] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Published online: 23 August 2016

(2) The need for customers to invest in website infrastructure and increased operating costs of such infrastructure management.
(3) Increased traffic density in the web originating from the data that is far from the user and also scrolling through all congested and analogous paths.
(4) Reducing the quality, speed and reliability of data delivery.
(5) Increased load of main servers.
(6) Creating delay or pause while hundreds of users choose a web page or data simultaneously.
(7) The possibility of non-secure failure, if any data is available on a server.
(8) Lack of inherent backup copy, archive and better capacity of content storage.
(9) Difficulty of managing the assignment of IP address to all computing devices and mobile phones.
(10) The difficulty of assigning IP addresses to mobile nodes.
(11) Need of mapping data for a specific node, resulting in low quality delivered.
(12) High control overhead for each hop of a routed path to identify the sender of the next hop.
(13) The possibility of sniffing when data is transmitted in wireless broadcast channels.

The named data network (NDN) tried to solve this problem by the routing mechanism using the name instead of an address. This proposed mechanism enables network to use saving pieces of content in the cache so as to optimize content distribution instead of communicating and getting permission from the routers.

In NDN, data are identified by networks with unique names and NDN packages including the names of the requested or carried data. Using these location-independent characteristics, network equipment and end-hosts know what they transfer. Also, they can inherently perform content-based operations. These operations include content routing (or paths by name), storing data packages in the cache, or network data processing. On the other hand, the flexibility of NDN can reduce transmission costs for the internet service providers (ISP), and therefore improves end-user performance and enhance quality of service on the network. In fact, supporting content-based functions at the network layer allows going beyond what is done today with content delivery. For example, if the network devices are aware of the content they carry, issues such as network neutrality, security and privacy can be created. Network routers must have additional operations on transition plans and such issues are raised.

In a network where there is no IP address, routing is certainly one of the main issues raised. Several algorithms have been proposed in association with this issue, some of which are reviewed in Sect. 3.

This paper presents a new routing protocol for NDN using learning automata and clustering. In this protocol, network is an undirected graph whose nodes are grouped in clusters. Cluster heads (CH) are responsible for storing the location of the named data and routing. Furthermore, they are equipped with learning automata and can find optimal Steiner trees using learning in order to have a multicast tree for routing. Although Dijkstra's algorithm can obtain the shortest path between graph nodes, the node address is in information-centric networking. Also, since nodes cannot identify each other, hence it cannot be easily used. In the proposed algorithm, paths are converging toward Dijkstra paths using learning and the shortest paths are found.

The correctness of the proposed algorithm have been proven by expressions and then implemented on a sample network using the NS2 simulator. Also, important parameters were evaluated and compared by changing the number of requests.

The second section of this paper provides the NDN and the relevant issues. This section is a basis to understand the following sections. The third section presents some conducted works in NDN routing. The fourth and fifth sections explain the background of learning automata and clustering algorithms of learning automata and clustering algorithms. More so the proposed protocol is investigated in the sixth section. In the seventh section, the proposed protocol is proven by expressions, the eighth section presents simulation results under NS2[1] environment. Finally, the ninth sections explain the complexity of the algorithm.

2 Basic techniques in NDN

This section explains how Content-centric networking (CCN) and NDN networks work. The information provided in this section is mainly a combination of a high level of NDN, which widely presents the sections provided on software for implementation in CCN (CCNx project) protocols [2].

2.1 Naming and routing

At NDN, when a node is interested in a certain content or information, it can send an interest package containing a descriptive name of the content. Each node that receives this package, sends it to the node that is responsible for the production of the information until the package reaches the producer or the node which supply the requested content from the cache.

[1] Network simulator.

When an interest package reaches such a node, a package containing a content object is created and is sent to the data requester through the reverse path and its interest package is discarded. Whenever a node finds content in accordance with the requested information in the cache along through the path which was maintained in the previous request, the forwarding interest package is stopped and the object sends its cache toward the requester. The purpose of cache mechanism is to reduce the workload on the information generating nodes that their data are frequently requested. The second request for the data that has been delivered recently can be supplied from the first node in the path between the requester and the producer [3].

Principle of the NDN states that it does not matter who is delivering the data, but what matters is that they should be valid, while in IP networks, nodes and links that are popular or highly requested have high overload. In NDN, the possibility that a node near us, on the path to content generator, has a copy of the required content in the cache, increases the popularity of this method. Through the mechanism of the cache, content copies are automatically distributed to parts of the network that are requested [1].

Information and content are located in the caches close to request locations. It can be said that the load on the network is reduced by pieces of content, when its requests are repeated and in this case, the nodes in this path can be used instead of the content generator server. This is a very useful feature for a network in which several organizations share relative equal information and requests for the information are too much [1,2,4].

Since routing and forwarding are done by name, a structured solution is required for the content naming. CCNx uses a hierarchical structure in which each name is a combination of several segments like CCNx:/First name component/ Second name component /Third name component. Each next segment or component describes a feature of content more accurately. Description of a name is not only that the nodes in the network must select forward rules submitted by the listed prefixes, but is the meaning of a name for the group requests and relevant information generated. Packages are sent via interfaces, which can be any type of connection with other processes, daemon, node or link. Once the prefix of interest is matched to a name of a content, the content is eligible for request, and can be returned to the requester. To extract the names of the content, the name should be developed by additional parts or components in order to show version and segmentation. For example, central components of ⟨part number⟩ and ⟨time stamp⟩ are defined to specify the segment and content version [2]. Since these components are added after the significant segments of a prefix, the added components must not disturb routing, but create communication between the user and application. There are important guidelines for content naming in the world as shown in the following [2,5]:

(1) Name of the content must be unique in the world.
(2) Name of the content must have a DNS name as the first component.

Like all network mechanisms, nodes or routers require a mechanism for decision-making on how to send packages across multiple networks. In CCN, all nodes (routers, servers, users and peers) are equal in performance; this means that they can send all requests, even service, and store the data in cache.

Each node maintains 3 tables (Fig. 1) and decides on what action to take for a content object or interest based on their content. These three tables are as follows [1,4]:

- **Content store (CS)** It is a temporary cache of content objects the router has received for a given period of time.It is used to temporarily store data packets, and the like can be the buffer memory IP-based routers. So, for reducing the amount of upstream flow bandwidth demand and delays in downstream flows, as long as possible, the NDN remembers the incoming data packet, because each NDN packet is reusable for several consumers.
- **Pending interest table (PIT)** It contains a list of incoming faces and interests for which the node has forwarded but not satisfied yet. Each PIT entry serves as waypoints for a content object together with its incoming and outgoing interface(s). This table holds interest packets being sent as upstream to the sources, who are yet to be answered. The returned data can be sent via the same route that leads to the requester as downstream. PIT entries are deleted as soon as the requested data are found, and if not found at the time specified, finally, the time expires.
- **Forwarding information base (FIB)** It contains forwarding rules for naming prefixes. According to this table, interest packets can be sent to a list of interfaces, which shows its difference with IP-FIB. This difference indicates that the request can be sent to multiple sources at the same time.

In the following, the function of each of the tables is explained further.

CCNx program determines the correct activity using the prefix matching of different components with different names and tables. Full prefix matching occurs in naming components, for example, ccnx:/Eli /photo as ccnx:/Eli tends toward name matching of ccnx:/Eli, while this name is not a prefix of it. When the name of a received interest prefix is matched to the name of a content object in CS, the node can discard inter-

est and return a copy of the information stored in the cache. A requested name may be matched with several names of prefixes, in this case, at least one of the content which satisfy the interest is returned [5,6].

If pending interest table (PIT) includes a precise matching, it means that the node currently has a pending request for the same information. Origin of the face of interest is added to the list of pending faces of interests to ensure that when a satisfactory response is returned, this request will be answered too. After being added to the list, the interest is discarded because the proper forwarding operation is running.

If interest has not been still answered, forwarding information base (FIB) is a good choice for search. If any of the prefix matching rules is matched with the incoming interest, interest is sent to the specified face in the rules which have the longest matching prefix with the name of interest [7].

Since face can be used as a link, a connection, tunnel or application, it is possible for interest to be sent to a node with a closer hop towards the generator or host application or content generator. When a node has the longest matching prefix with several rules, interest can be repeated and be sent to several faces. A rule is added in the PIT table which shows that the interest has been sent and shows the origin of the face. Chain of the rules in PITs on all nodes will be from the requester to generator which are used to send a content object to requesters based on the reverse path. The process of determining how to process an incoming interest by name is shown in Fig. 1 by monitoring the content of CS, PIT, and FIB [8].

When a content object enters a node, for each PIT entry (prefix) that matches with the content, a copy of the content object is sent to all requested faces and entries are cleared from the table. In each subsequent node which receives a copy of the object content, this process will continue until the chain achieves the aim of the requester, and finally, the content object is presented to the requester application [9].

2.2 Strategy layer

One of the new options in NDN is a strategy layer. The philosophy behind the strategy layer, is that a process must be monitored on all objects in the router, and change forwarding decisions based on record events. CCNx now follows strategies for all node runs. Incoming interests are sent to all interfaces for finding the longest matching prefix, based on interest submission. A node will periodically retransmit interest from the active entries in the PIT. It is recommended that this should be retransmitted at random times in a different interface, or retransmitted using a heuristic scheduling [1,3,5].

2.3 ISP-based aggregation

One method for long-term routing is ISP-based aggregation, which can dynamically generate names. The proposed solution is to differentiate between names chosen by the user and provider name assigned. These two names are mapped using service analogous to DNS. Names assigned by providers are built hierarchically, where long names define an exact location on the network [2].

2.4 Cache organization

Content management plays an important role in the performance of NDN, and it also depends on the method of caching on these systems. The cache organization consists of caching methods and updating the contents, to ensure freshness, accuracy and health of the content. Cache organizations could

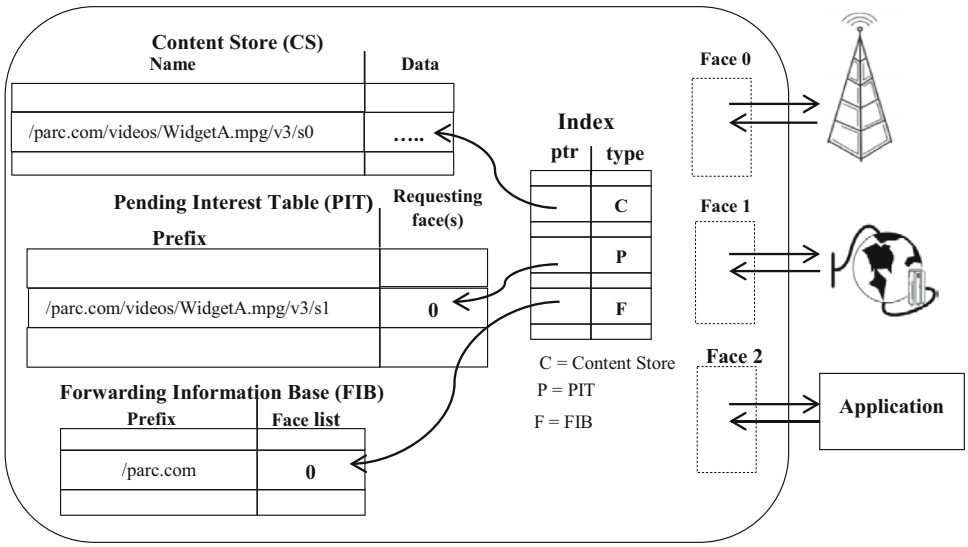


Fig. 1 Packet forwarding process at an NDN node [1]

include integration and linking of caching systems. So this integration also affects the management of content in these networks. Improvement of the efficiency of a content distribution network, which has been set by caching and replication together, is measured based on the delay of the hit ratio and byte hit ratio. However, repeated use of caching and replication together in these systems makes them resistant against the flash crowd events [7].

2.5 Security

One of the CCNx major concerns, is providing security solutions for high-value and confidential content. Security means protecting content against unauthorized access and modification. Without proper security controls, CCN platform exposes internet fraud, distributed attack, viruses or other unwanted intrusions, that can cripple the business. The NDN contrary IP, do not use encryption of transport tunnels from the requester to the generator. So there is a need to change the current public key infrastructure, to ensure security control [4].

The summarization algorithm SHA256, is used now for the last segment of the name, for each content object, to ensure that the full name is addressing a unique content. In CCN, all content authentication of digital signature and private content are protected with encryption. Embedded security on the content and not on the host, will reduce the need to trust in the interface location, and increase wider participation in the network [1].

3 Related works

As previously explained, in NDN, the user sends interest packages along with the names of data to obtain data. Since interest messages do not include source and destination addresses, routers should forward the packages based on the names they carry. Routing protocol in vast NDN networks should be based on the name and for this purpose, it requires calculating and installing the appropriate entries in the table of NDN forwarding nodes (FIB). Each FIB entry contains a prefix and one or more next hops, used to forward interest packages whose names are matched with the prefix; IP is only used for one and the best next hop or several paths with equal costs to avoid loop creation. NDN can freely use multiple paths, because prevention of loop creation is conducted in its forwarding process. So, NDN network needs routing protocol which supports name-based multiple routing. Table 1 represents some routing protocols for NDN networks and some protocols provided for information-centric networkings.

4 Automata

The learning process of living creatures is considered as one of the new research topics. This research is divided into two general categories. The first category examines the learning principles of living creatures and its steps, and the second category seeks to offer a similar methodology for putting these principles in a machine. Learning is defined as changes created in the efficiency of a system based on past experiences. An important characteristic of learning systems is the ability to improve their performance over time. Mathematically, it can be said that the purpose of a learning system is to optimize the task that is not fully known. Therefore, one approach to this problem is to reduce the purposes of learning system to an optimization problem which is defined on a set of parameters and aims to find a set of optimal parameters [23]. A learning automata is made up of two main parts [23,24]:

I. A stochastic automata with a limited number of actions and a random environment that the automata is associated with.
II. Learning algorithm that the automata uses to learn optimal action by using it.

4.1 Random automaton theory

A stochastic automata is defined as a quintuple $SA = \{\alpha, \beta, F, G, \phi\}$ and r is the number of automata actions, $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the automata set of actions, $\beta = \{\beta_1, \beta_2, \ldots, \beta_r\}$ is the automata inputs set, $F = \phi \times \beta \rightarrow \phi$ is the production function of the new state, $G = \phi \rightarrow \alpha$ is the output function which maps the current state to the next output and $\phi(n) = \{\phi_1, \phi_2, \ldots, \phi_k\}$ is the automata inner states set at the moment n [23,24].

Learning automata is categorized into two groups: fixed structure automata and variable structure automata. The stochastic automata with the fixed structural possibility of actions are fixed. While the random automaton with variable structure of the possibilities of actions in each iteration is updated. On learning automaton with variable structure, change of the possible action is based on a learning algorithm. However, the internal state of the automaton's ϕ sets is represented by the possibilities of actions. In fact, the automaton as a state-output automata considered that its output is equivalent to its internal state. The internal state of automaton $\phi(n)$ at instant n, with probability set of actions P (n), is shown as follows:

$$P(n) \equiv \{p_1(n), p_2(n), \ldots, p_r(n)\} \text{ so that}$$
$$\sum_{i=1}^{r} p_i(n) = 1, \forall n,\ p_i(n) = \text{Prob}[\alpha(n) = \alpha_i]$$

Table 1 Some protocols conducted in NDN and CCN

Protocol name	Main features of protocols
NLSR [10,11]	Using two types of LSA adjacent and prefix; full recognition of topology by each node by declaring LSA; storing tables called LSDB; synchronization of LSDB periodically and hop-by-hop and not torrential; carrying signature in each package; hierarchical naming; using Dijkstra's algorithm to find the shortest path; multiple routing; finding several next hops in the process of routing
CRoS [12,13]	Dividing the network elements into two roles of router and controller; Router: forwarding packages to the destination and maintaining the names of data; Controller: calculating and storing the named data locations; obtaining topology at Bootstrap phase and calculating some paths to all routers; Bootstrap phase: finding the registered controllers by routers; discovering the topology by the controllers; installing paths to the routers by controllers; Routing: registering the named data; installing the path; finding the path
OSPFN [14]	Maintaining the information on links in the LSDB by all routers; using OSPF to find the shortest path; updating LSDB when link state announcing (LSA) is flooding; saving a copy of LSDB in all routers of the network and building a network topology in them
Two-layer intra-domain routing scheme [15,16]	TM topology storage layer: topology discovery; management of node or link error; calculation of the shortest path tree; PM prefix announcement layer: active propagation and passive service of prefixes according to scalability; categorizing routers to content requester and provider; using two functions of announcement forwarding and FIB making; Maintaining a complete topology of the network in each node through torrential propagation; Providing content by active propagation and passive service
Adaptive forwarding [17,18]	Introduction of NACK Interest; a ranking of interfaces in FIB; coloring schemes of interfaces and forwarding strategies based on the coloring (green: interface works, yellow: interface might work and might not work, red: interface does not work); locating a rate limit on an interface using the link capacity; estimated package size error parameter; forwarding strategy: from the most prioritized green interface and if absent, the most prioritized yellow interface
SIR [19]	Adding two fields of delay and the number of hops in Interest package. Distributed function (no need to fully understand the topology by the nodes). Conducting the main steps in two phases of Interest and content propagation. Calculating the shortest package based on the delay field in receiving Interest package
CCNFRR [20]	Detection of a node or link failure and informing the adjacent without torrential propagation. Defining FIB and PIT packages for announcement of failure to the adjacent and having an alternative path
Cluster based multipath routing [21]	Adding Condition Database (CDB) to set the load, saving the number of data packages in CDB by cluster head, using CDB for restoration of Interest package, changing the crowded path by Interest package restoration
Probabilistic ant-routing mechanism [22]	Interest: The ants who are looking for food, data packages: the ants that have food; pheromone: PIT entries; adding a quality field to FIB in order to show how Interests are conducted, editing FIB entries and their quality field when data packages and Interest achieve

At the beginning of the automaton, the probability of the actions is equal to $1/r$ (where r is the number of actions).

4.2 Environment

Every environment is demonstrated by the triple $E = \{\alpha, \beta, c\}$, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ set of environmental inputs, $\beta = \{\beta_1, \beta_2, \ldots, \beta_r\}$ set of environmental outputs, and $c = \{c_1, c_2, \ldots, c_r\}$ is the set of the probability of penalties [23].

Input of environment is one of the r selected action. Output (response) of environment to any action i, will be determined by β_i. If β_i is a binary response, the environment is called P-model. In such an environment, $\beta_i(n) = 1$as an unfa-

vorable response or failure and $\beta_i(n) = 0$ as a favorable response or success are considered. In the environment of Q-model, $\beta_i(n)$contains a limited number of values in the [1, 0]. Whereas in the S-model, values of $\beta_i(n)$are a random variety of [1,0] ($\beta_i(n) \in [0, 1]$). The set c, defines the penalty possibilities (failure) of environmental response and is defined as follows:

$$c_i = \text{Prob}\{\beta(n) = 1 | \alpha(n) = \alpha_i\}, \quad i = \{1, 2, \ldots, r\}$$

This shows that the probability of α_i action gets unfavorable response from the environment. The α_i values are unknown and it is assumed that they have a unique minimum value. Similarly, the environment can be shown by the $\{d_i\}$ set of probability reward (success), that in this case, d_i indicates the possibility of receiving favorable response to the α_i action. In stationary environments, values of penalty probability are fixed. Whereas, in non-stationary environments, values of penalty probability will change over time [24].

Figure 2 describes the working principle of the learning automata. The collection, along with the learning algorithm, is called Stochastic Learning Automata. Stochastic learning automata can be shown in quaternary $LA = \{\alpha, \beta, p, T\}$ where r is the number of actions, $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of actions, $\beta = \{\beta_1, \beta_2, \ldots, \beta_r\}$ is the set of inputs, $p = \{p_1, p_2, \ldots, p_r\}$ is the vector of the probability of actions and $T = p(n + 1) = T[\alpha(n), \beta(n), p(n)]$ is the learning algorithm.

In the environment of Model S, the environment is defined as follows:

$$E = \{\alpha, \beta, s\} \text{ that}$$
$$s(n) = \{s_1, s_2, \ldots, s_r\}; s_i = E\{\beta_i(n) | \alpha_i\}; \forall i$$

S_i is the amount of β_i responses mean for the action α_i, and in fact, s_i is considered as the severity of penalties. It is presumed that in the n-th hop, α_i action is selected, and $\beta_i(n)$ is the response of the environment. In that case, the possibility of automata actions of $S - L_{RI}$is updated as follows.

$$\begin{aligned} p_i(n + 1) &= p_i(n) + a(1 - \beta_i(n))(1 - p_i(n)) \\ p_j(n + 1) &= p_j(n) - a(1 - \beta_i(n))p_j(n) \qquad \forall j, j \neq i \end{aligned} \tag{1}$$

a is the learning parameter, and $0 \prec a \prec 1$.

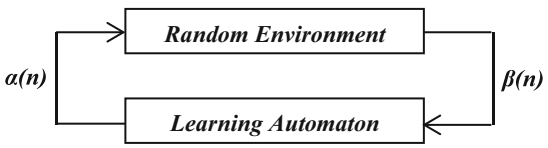


Fig. 2 Learning automata

In the environment of Model P, the environment is defined by the possibility of penalties. For every action such as α_i, the environment responds to the automata with a random value of $[\beta_i(n)|\alpha_i]$ which forms the input of automata. In Model P, $\beta_i(n)$ response to the possibility of c_i and $1 - c_i$ are considered equal to 1 (poor responses) and zero (favorable response), respectively [23]. In S environments, environmental response to learning automata actions is a random variable on the interval of [0, 1].

5 Clustering

Classification of network nodes in a number of virtual groups, to reduce the routing overhead such as applications in the network, is performed by conventional methods known as network protocols. Hence, this method is called clustering [25]. In clustering algorithms, the network is divided into a number of virtual groups such that each of these groups are called clusters. Nodes in each cluster are geographically adjacent to each other. In each cluster, a node is selected as the cluster head (CH). Other nodes are members of a cluster. The maximum distance from any node to its cluster head, is called the cluster radius. Cluster radius is one of the important parameters in designing algorithms for clustering and is expressed by the number of hops. Most algorithms are provided by 1 hop, that is, cluster members are located at a distance of one node to the cluster-head. Some algorithms are also multi-hop. The proposed algorithms have different methods for selecting cluster heads. Due to the limitations capacity of nodes in the network, in usage such as routing and distribution of information, use of hierarchical structure to optimize network capacity, is necessary. In fact, clustering is a way of achieving such hierarchical structure [48,49].

5.1 A review of clustering algorithms

Before introducing the algorithm used to perform clustering in the proposed protocol, an overview of the clustering algorithms was given [48]. The most clustering algorithms are the lowest-ID and max Degree, that are the basis of most other algorithms. In the lowest-ID from each node, ID is used to identify the cluster head. Since clustering is implemented at the network layer, the ID of each node and the node IP address, which is a 32 bit number are considered. Lowest-ID is a 1-hop algorithm. Consequently, the radius of clustering is considered as a node. In this algorithm, each node has the lowest ID among its neighbors, and the cluster head. In the max-Degree, the number of neighboring nodes is the main decision-making parameter. Thus, the node is selected as a cluster head that have more than the required number of neighbors. A neighboring node number, is a measure of connectivity. On the other hand, for clustering algorithms,

the ideal dominating set (DS) is used. As defined, in graph G, a set of nodes called S are a DS, such that each node of the graph G is a member of S or one of the members of S neighbors. In fact, this definition explains the 1-hopDS and also extends to explain the d-hopDS. One clustering algorithm called max-min is provided based on the d-hopDS.

There are other algorithms, but the basic ones are the lowest-ID and max-degree. By doing simulation on both models, lowest-ID and max-degree, and evaluating the results, it has been shown that the lowest-ID algorithm is more stable than max-degree. In other words, the number of changes in the cluster heads in the lowest-ID is less than the number of changes in the cluster heads in the max-degree. As a result, the lowest-ID algorithm is used in the proposed protocol for selecting cluster heads.

5.2 Advantages of clustering

In clustering, cluster heads of each group will be responsible for routing within the group. The gateway nodes exchange data between two adjacent clusters. Therefore, cluster heads and gateways constitute the virtual backbone. This brings the following benefits:

- Number of control packets in the network decreases and there is limited use of network capacity. This will reduce power consumption in mobile nodes. In addition, all mobile nodes are not required to keep routing tables. Thus, there is reduced usage of network resources.
- Mobility nodes will have less effect on performance routing. Each node's routing information are only stored in the cluster heads of their cluster. By moving a node from one cluster to another cluster, it was observed that only cluster heads change.

6 Proposed protocol

In this section, the proposed algorithm is examined to provide a new routing protocol using learning automata and clustering in NDN.

6.1 Naming

Perhaps the most important part in the design of a protocol for NDN is providing a proper naming scheme for each element in the routing system [1]. Based on the current network structure and operating practices, it seems that a hierarchical naming scheme is appropriate to show the relationship between different components of the system, so identification of routers belonging to the same network and messages produced by the routing process is easy. In the design presented, all nodes, including end nodes, routers and cluster heads are named according to the network in which they reside, and the sites, and also nodes' own names [28]. In the suggested naming, fixed prefixes are used to display the type of node to make routing easier. Routers and clusters are specified by /Router/ and /Cluster head/ prefixes. Some prefixes are intended to facilitate routing, for example, data and interest packages should be distinguished from routers and cluster heads recognizing packages as well as the packages that determine the cluster heads. It should be noted that intermediate components of network, all elements except end nodes, have a unique ID which can be identified by that, and this name or ID has nothing to do with their NDN [32].

6.2 Topology recognition

After running the algorithm of cluster head selection, cluster heads forward interest packages to their own cluster members to introduce themselves to them and also obtain information from the nodes [1,2]. Cluster heads collect the information of their cluster nodes and store the information in the table of members and form the clusters [30]. Each node also maintains a cluster membership table that contains the cluster of the node in the network, which indicates that a node can be a member of more than one cluster [33]. The most important information that the cluster heads of the first level obtain include the location of named data, so that each end, or generator node, should register its named data in the relevant cluster head. Of course, registering of the named data is not only done at this stage; whenever a producer has an unregistered named data, it should inform the cluster heads and register the data on them [31].

Each router that receives an interest package forwarded by a cluster head, answers with a package of data, including ID, sequence number, the links and its adjacent. When the cluster head receives a data package, it registers an entry to identify the router in FIB and update a table of its members. Whenever a node observes changes in the link state, it must announce the differentiation to the cluster head so that the cluster head would have accurate and precise information about its members [34].

Routers must know each other to be able to route packages toward their cluster heads, for this purpose, routers periodically forward interest packages in all interfaces. Each router that receives the package, responds with a data package containing ID and ordinal number, so each router has an updated list of its adjacent [34].

In addition to recognizing cluster members, cluster heads must also recognize adjacent cluster heads. For this reason, each node saves a table of adjacent cluster heads and addresses and puts the addresses in forwarding interest packages. Thus, the cluster head receives interest messages from its adjacent nodes, and gets informed from adjacent clusters.

At the time of route request of the source node, the cluster head is responsible for distribution of interest packages to adjacent cluster heads. In the interest packages while passing through the nodes, the name of the elements of the route should be included in the package and only the name of the cluster heads which the request package passed through is recorded.

As mentioned in the next section, in our proposed architecture, clustering is done hierarchically; at any high level of clustering, a summary of information about lower cluster heads must be available. Cluster heads are members of a higher level cluster and they exchange the information of their links according to the summarized information on the lower level. A node at any level, forwards the information obtained at that level after the implementation of the algorithm to its lower level. The lower level also has hierarchical topology information. Each node has a unique hierarchical name; there is a way to correspond with the hierarchical name of the cluster number, starting from the root.

After topology phase recognition, all routers recognize cluster heads. Cluster heads have topology and are able to find routes between routers, and most importantly, cluster heads know the location of named data and provide this information hierarchically to the cluster heads of higher levels. As a result, there is a complete knowledge of the network topology which allows routing of customers' requests toward content providers.

6.3 Learning automata in the proposed protocol

The proposed protocol is an intelligent protocol whose main idea is based on machine learning algorithms especially learning automata. Since the IP address is not available on NDN networks and the nodes must somehow know each other, our proposed protocol, which will be called LCRN (Learning automata and Clustering based Routing algorithm for NDN) from now on, uses learning for this recognition. For this purpose, reinforcement learning of $S-L_{RI}$ was used in the proposed algorithm, which was described in Sect. 4.2.

Since LCRN has been suggested for a dynamic network like NDN, P_model automata cannot be used, because they have weak convergence in dynamic environments. Reinforcement learning of $S - L_{RI}$ is used among the various options available on the S model. The reason for this choice is the lack of direct penalty for the automata that have not been selected in this scheme [40,41]. Since the network is dynamic and conditions are changing, if some paths are penalized, the possibility of their selection in the future is zero. However, we know that for the change in queues and buffers, the path which is not a good path now, may later turn into a proper path. Consequently, the chances of their choice should not be eliminated by permanent penalization in the future [42].

6.4 Clustering in the proposed protocol

In the proposed protocol, the network is considered as an undirected graph where the nodes are grouped in clusters. Cluster heads (CH) are responsible for storing the location of named data and routing. The cluster heads first obtain the network topology and they also know the other cluster heads and keep their information. In the proposed architecture, the network elements have two roles: router and cluster head. The router is the main element of the network that is responsible for leading the packages to the destination and recognizes the other cluster heads; and cluster heads are responsible for calculating the route and storing the location of named data. In the stage of topology recognition, cluster heads must be determined first and then routers and cluster heads must recognize each other. These procedures will be explained in the following sections [25,48].

As will be explained in the next section, the architecture used for naming is hierarchical, and it also helps hierarchical clustering. The proposed protocol uses the name of the site and network for naming, so these divisions make up the first and second levels of clustering, that is, the site, is the first level of clustering and the network is the next level of clustering [26]. Using hierarchical routing, network nodes are divided into some groups. The roles played by nodes are not the same in routing. In each group, one node is responsible for keeping routing information [27,29]. As a result, routing is done with nodes through a network. Considering the hierarchical routing, a virtual network can be assumed, whose backbone members are responsible for routing. Thus, hierarchical routing algorithms reduce control of exchange traffic and thus increase network capacity (view of clustering in LCRN algorithm is presented in Fig. 3) .

As mentioned, clustering in LCRN is based on the naming of the data. Any data that is generated, is named based on the two levels of its location, that is, a mean site and network. According to the local level, clusters are created. As a result, if the data is distributed and repeated in other places, there is no need to rename or change the configuration of the clusters. The main purpose of creating clusters in the LCRN, is to prevent the spread of routing overhead among all nodes. Each cluster heads have the task of searching the data in cluster members [48]. Since in the topology recognition phase, cluster heads within the neighborhood know each other, a large amount of routing overhead is reduced. On the other hand, in LCRN all nodes are not required to keep the network topology information, and it saves the network resources. In the proposed protocol, the third level of clustering is also considered. The objective from the third level, is to aggregate and manage multiple network with each other, and reduce the workload routing. In terms of prefix matching, in LCRN, clustering has no interference to do prefix matching, and matching is done based on the named data of

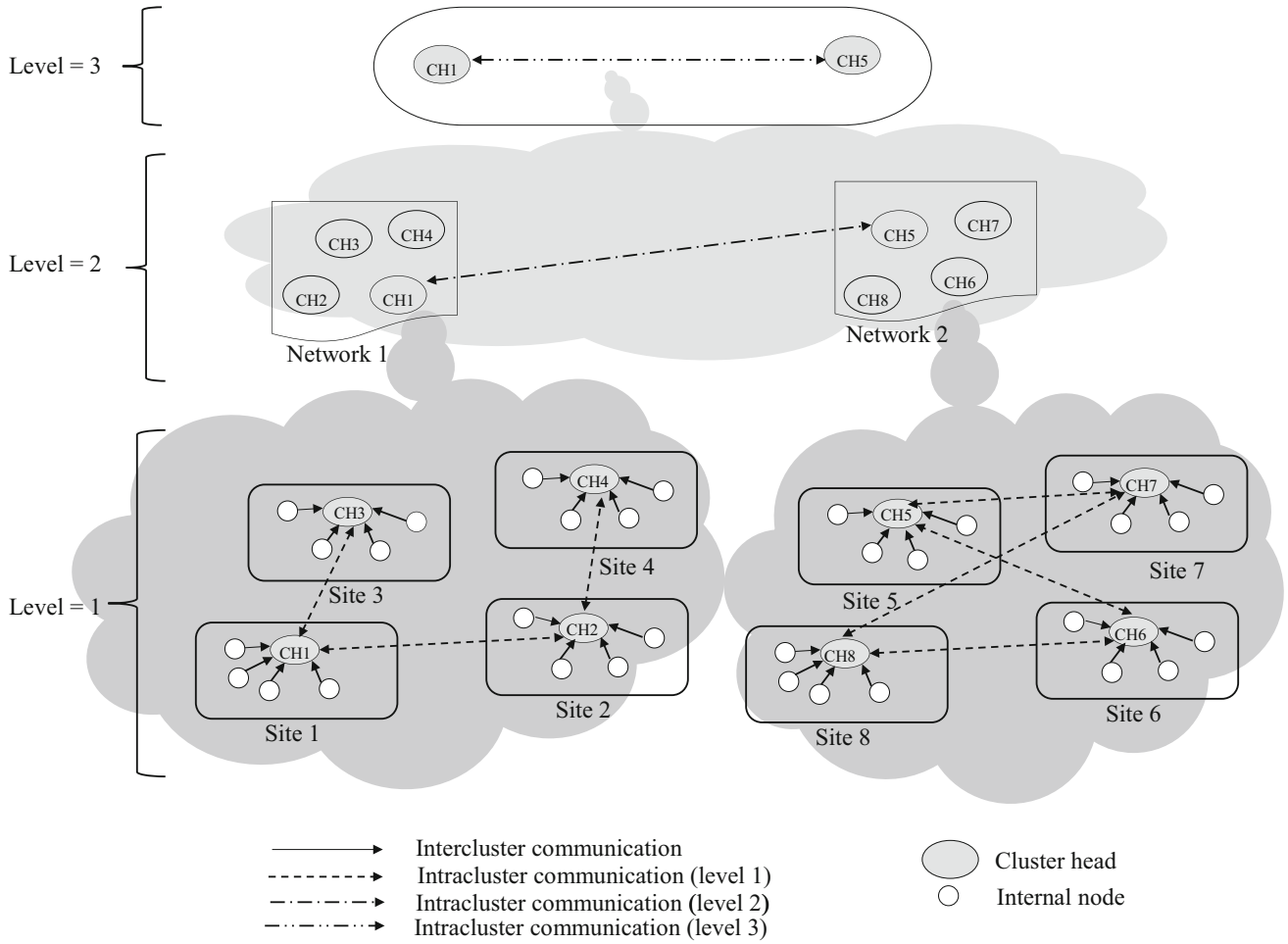


Fig. 3 View of the clustering in LCRN algorithm

every cluster, and also the data that is stored in the cache of cluster heads. So clustering performed in LCRN, such as in accordance with the naming data, is inconsistent with the decentralization of information on the NDN [49,50].

6.5 Routing tables

In LCRN, all routing tables are only located in cluster heads and the end nodes lack these tables. According to NDN rules, each node keeps 3 tables as follows, that some changes have been created in some of them:

- **(CS) and (PIT)** These tables are used in LCRN protocol with their standard format in CCNx.
- **(FIB)** Table 2 represents the structure of this table in LCRN.
- **Dijkstra's table (DT)** It is another table which is added in order to evaluate the performance of learning automata. This table is stored in cluster head nodes and the records in each of them represent the smallest cost of this cluster head to each of the other categories, and the fields of the table are route, route length and route expenses using Dijkstra's algorithm.

Table 2 FIB table structure

Field title	Field description
Prefix	Name prefix
Face	An interface where the package of this matching prefix comes from
q	The possibility of achieving the name prefix via this interface
Dijksrtra_record	Boolean field which indicates that the corresponding record is recorded through learning algorithm or Dijkstra's algorithm

6.6 Dijkstra's algorithm

Dijkstra's algorithm is a greedy approach for finding the shortest path from a fixed destination (single source) to other weighted graph nodes. Naturally, this algorithm is applied

to find the shortest path between two nodes. The only condition to use this algorithm is non-negative weight of graph edges. In topology recognition phase, LCRN algorithm finds the shortest path between each cluster head to other cluster heads and records them in a table called Dijkstra's table (this table will be updated periodically). As will be explained in the next section, the purpose of this table is to review learning automata performance.

We know that Dijkstra's algorithm gives the shortest path between two nodes, but in NDN network, since nodes do not have IP addresses and are not identifiable for other nodes, we cannot use this algorithm. In this regards, after running LCRN algorithm and finding nodes having content, the shortest path can be achieved based on Dijkstra's algorithm and consider it as a proper response of the operating environment to automata [35].

6.7 Routing

When each node asks for a named data, it forwards an interest package to the network and specifies the name of the data in the package: *Interest ("/Wanted Prefix")*
The first router that receives this package adds an entry to its own PIT; if there is no law in its FIB, the route should be discovered. Then the router creates a route request interest package with a significant name and forwards it to the first cluster head:
Interest ("/Cluster/ Cluster Head ID/ Route From/ Source Router ID/ Wanted Prefix")

Where it includes *"/Route From/ Source Router ID"* the source router that has sent the interest and *"/Wanted Prefix"* a recorded named data that the cluster head knows as the destination. Then the cluster head must find the best route from source to destination and receive a route response data package, that the proposed algorithm is used for this purpose.

6.7.1 Routing algorithm

The purpose of this protocol is to choose a path with the lowest cost in order to enhance the quality of Internet services. This is done using learning automata with multi-level clustering. In general, LCRN acts so that each node head at any time selects the route with the lowest cost using its own learning automata among the different routes (we remember that alternative routes should be found, according to existing rules in NDN). If the chosen path is the right path, it will be rewarded, otherwise it will be punished. The purpose of punishing and rewarding is low and high probability of their choice.

We know that according to NDN rules, alternative routes should be found for a route request and since caches in routers play an essential role in NDN and there is no need to provide the requested data from a specific point and only the named data is important; we use multicast trees to implement the transmission. One of the most common ways of constructing multicast trees is to use Steiner tree construction algorithms in graphs.

Steiner tree in the graph is defined as follows: consider $G\ (V, E)$ graph where V is the set of nodes and E is the set of graph edges [38]. A cost of $c(i, j)$ is assigned to each edge of (i, j) in the graph. A subset of nodes called T is also defined as a set of terminals. The purpose of the Steiner tree is finding a tree in the graph that includes a set of terminals with a minimal cost. This tree is called *a minimum Steiner tree*. If the tree obtained includes only terminals, it is called the minimum spanning tree and there are multiple polynomial algorithms like Prim's algorithm used as a solution; however, in general, non-terminal nodes (the nodes that are called Steiner points) are used to reduce the cost of the tree [36,37].

In LCRN, the purpose is to find the least expensive routes from content requester node to content owners, where we map this problem to find the shortest multicast tree where the tree roots are content requester, and the tree leaves have content. The multicast tree is also mapped to find the Steiner tree roots where the terminals have content and the root is content requester. Of course *RTT time* constraint should also be considered. It means that the purpose is to find the minimum Steiner tree at *RTT time*, that is, every terminal node that is covered is acceptable. The presented algorithm is an iterative algorithm [39]. In each iteration, a Steiner tree is made with the mentioned conditions. Each iteration starts when a route is requested by a content requester.

In LCRN protocol, each node has a learning automata with operations equal to adjacent nodes, which is used by protocols to choose the right path for route cost reduction. When the route request interest package reaches the first cluster head, it selects a route to forward the package, after recording information in interest package fields and according to the learning automata. Higher-level and intermediate cluster heads, which receive the interest package, forward the package to the requested name prefix according to the learning automata category, and cluster members table and also the information of the named data. Each cluster head that has a prefix matching with requester name, announces this matching and forwards it to the data requester using the inverse path mentioned in the interest package. To select the best route, rewards and penalties by automata, standards or criteria should be adopted. Delay, delay variation, queuing, bandwidth and package loss probability are the network parameters that are used more in routing [43]. Delay is a critical parameter in quality of service (QoS) for most applications, especially in interactive and multimedia programs, because these programs need the least delay to communicate with users.

Table 3 Selecting a link due to a combination of fuzzy parameters, delay and queue length

Queue Length Delay	High	Medium	Low
Low	Medium	High	Very high
Medium	Low	Medium	High
High	Very low	Low	Medium

Considering these conditions, delay is an important parameter that should be considered. We know that low delay in a path does not necessarily mean a better path, and does not reflect a higher bandwidth or better link productivity. Delay on a link at a moment depends on the traffic passing through the link, and a link with high delay may be at its traffic peak, and become a link with low delay in another moment [44]. So it is better to use other criteria along with the delay criterion for the selection of the route with the least cost. It seems that after the delay, average queue length in routers is one of the most important parameters that directly affects the time delay created. Loss of packages occurs when the queue related to the router is crowded and overflowed. As a result, if a link has a smaller queue length and delay, it has a higher priority in routing. Table 3 indicates the selection probability of a link according to a fuzzy combination of two parameters of delay and queue length which is called DQ. In LCRN algorithm, solving Steiner tree with random weights obtained from DQ is considered and there is no basic knowledge of the parameters. Pseudo code of LCRN algorithm is presented in Fig. 4, and the proposed algorithm described can be summarized in Fig. 5.

6.7.2 Calculation of dynamic threshold

If ST_k is a weighted Steiner tree obtained in the k-th stage for a name prefix, the shortest path from the root to the leaves is calculated and located in ξ_k. Then the shortest path between the requester node and content providers (the terminal nodes in the tree weighted Steiner (ST_k)) is extracted from Dijkstra's table and placed in π_k. The lowest amount of ξ_k and π_k is the threshold of k-th stage which is denoted by T_k. Now the desired prefix must be placed in the FIB table, and if this code already exists in the table, showing a probability field of (q), the records are updated. These records lead to learning automata in LCRN algorithm and makes it converged after application of several stages of automata to the routes obtained from Dijkstra [45].

6.7.3 Editing of activities probability vector

If the cost of a route obtained from LCRN algorithm was lower than Dijkstra cost, activities selected by the automata of the route obtained from LCRN algorithm will be rewarded, and the responses that the relevant automata receive from the environment will be the route cost of ξ_k, otherwise the activities selected by Dijkstra route automata are rewarded in which the response received from the environment is the cost of route π_k [45,46]. Distributed automata is used in this algorithm, and each automata edits its action probability vector by $S - L_{RI}$ learning plan. Active activities are deactivated again and probability vector will be edited as it was explained in Sects. 4.1.

7 Proving the proposed algorithm accuracy, using expressions

In this section, we show that the LCRN algorithm is converged to optimal Steiner tree with the minimum weight expected. These results show that the optimal solution can be reached by choosing an appropriate learning rate for the algorithm. Before proving we need some basic definitions that are mentioned below.

$p_i(k)$: Automata action probability vector
$K(k)$: Total probabilities calculated as $K(k) = \sum_{\alpha_i \in A(k)} p_i(k)$.
τ_k: Steiner tree created in stage k
ξ_k: Weight of the shortest path from the root to the leaves of the tree τ_i calculated as $\xi_k = Min(\sum_{\forall e_{(s,t)} \in \tau_k} W(e_{(s,t)}(k)))$.
μ_k: Weight of the shortest path from the root to the leaves of the tree τ_i according to Steiner table
π_k: Dynamic threshold calculated as $\pi_k = Min(\mu_k, \xi_k)$.
$W(e_{(s,t)})$ is a positive random value with unknown probability distribution and $W(e_{(s,t)}(k))$ represents the weight at stage k.
$\lambda_{(i,j)}$: Connected path from node v_i to node v_j
p_j^i : Selection probability of the edge $e_{(i,j)}$
q_j^i : Selection probability of the route$\lambda_{(i,j)}$
$a_{e(i,j)}(k)$: The general rate of constant learning which is based on learning parameter and random weight variance and is obtained as

$$a_{e(i,j)}(k) = \frac{a}{\sigma_{e(i,j)}(k)} = a\left[\sum_{k=1}^{k_{e(i,j)}} \frac{[x_{e(i,j)}(k) - \overline{x}_{e(i,j)}(k)]^2}{(k_{e(i,j)} - 1)}\right]^{-1/2}$$

where $k_{e(i,j)}$ represents the number of times that the edge $e_{(i,j)}$ is observed and $x_{e(i,j)}$ and $\overline{x}_{e(i,j)}$ show the edge weight of $e_{(i,j)}$ at stage k and average weight, respectively. It should be noted that at the beginning, this variance must have a value in the range of (0, 1) for every edge [35].

Theorem 1 *If $q_i(k)$ is the probability of making τ_i tree at stage k, if $\underline{q}(k)$ is edited according to the proposed algorithm,*

Algorithm LCRN

1: Input: Stochastic Graph $G<V,E>$, Customer Node with name "*s*", Interest ("/*Wanted Prefix*"), *RTT* time for Interest Packet
2: Output: Minimum Steiner Tree to Named Data with *Wanted Prefix* (Producer and Cache Memory)
3: Assumption
4: *Let k be the iteration number and initially set to zero*
5: *Let* ST_k *be the Steiner Tree selected at iteration k and initially set to* $\square$
6: *Let* ξ_k *denotes the minimum path from root to leaf in be the Steiner Tree selected at iteration k*
7: *Let* D_{ij} *denotes the Dijkstra table in the cluster heads at iteration*
8: *Let* μ_k *denotes the Min path from root to leafs in* ST_k *from* D_{ij}
9: *Let* π_k *denotes the dynamic threshold of iteration k*
10: Begin Algorithm
11: Assign an Automaton to each Cluster Head and initially set it to passive state
12: $\pi_k = 0$
13: $k = 1$
14: s calls procedure **LR**(CH_i,k)
15: ξ_k = Min path from root to leafs in ST_k
16: μ_k = Min path from root to leafs in ST_k from Dij
17: π_k = Min (ξ_k, μ_k)
18: **If** $|\xi_k| \leq |\pi_k|$ **then**
19: Reward the selected actions of activated automata in ST_k with $\beta = |\xi_k|$
20: Penalize the selected actions of activated automata in μ_k
21: **Else**
22: Reward the selected actions of activated automata in μ_k with $\beta = |\pi_k|$
23: Penalize the selected actions of activated automata in ST_k
24: *k*=*k*+1 for *Wanted Prefix*
25: π_k = Min π_k and ξ_k
26: s starts to receive Data packets along π_k
27: End Algorithm

Procedure LR(CH_i,k)

1: Input: LR message
2: Assumption
3: *Let* CHS_i *be the Cluster Head Set of cluster head* CH_i *at iteration k and initially set to* $\square$
4: *Let LA denotes the set of activated automata which is initially set to* $\square$
5: *Let* α_j *denote the set of actions that can be taken by learning automata* A_j
6: *Time=0*
7: Begin Algorithm
8: **repeat**
9: **If** Prefix Matching with members in CH_i **Then**
10: $ST_k = ST_k + CH_i$
11: Trace back and Send Data Packet to s with name Reply
12: **If** $|\alpha_i| \neq \square$ **Then**
13: **For** all Next Level Cluster Head that not in ST_k **Do**
14: Send Interest to Next Level Cluster Head To find Prefix Matching (say CH_j)
15: CH_j with minimum cost and better Prefix Matching in its cluster heads select one (say CH_i)
16: **Until** (time < RTT)
17: End Algorithm

Fig. 4 Pseudocode of LCRN algorithm

then there is a learning rate of $a^*(\varepsilon) \in (0, 1)$ *for every* $\varepsilon \succ 0$, *and for* $a \in (0, a^*)$, *we have (theorem 1 in [35]):*

$$prob[\lim_{k\to\infty} q_i(k) = 1] \geq 1 - \varepsilon$$

Proof The proof of this theorem is carried out in various stages. At first, we prove that if k is sufficiently large, the possibility of tree penalty will converge to a fixed amount of final penalty probabilities. This feature is shown in Lemma 1. Then this indicates that the probability of selection of a tree with the lowest weight expected, is a process of *sub-Martingale* for large values of *k* and therefore the changes in construction possibility of a minimum tree is always non-negative. Lemmas 2 and 3 shows this result. As a result, the convergence

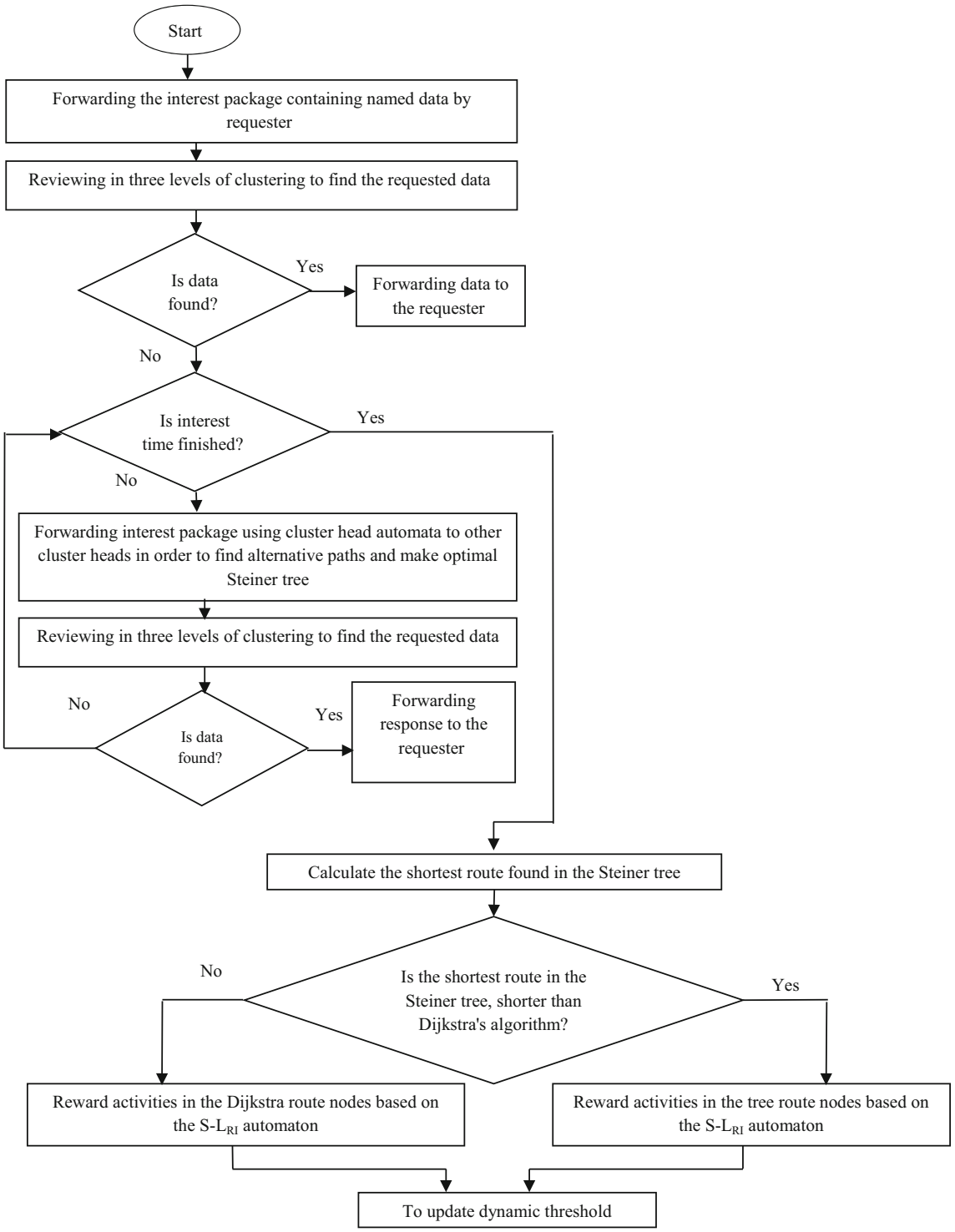


Fig. 5 The flowchart of the proposed algorithm

of the proposed algorithm to the tree with the lowest weight expected is proved by the convergence theorems of *Martingale*. Hence, to prove Theorem 1, the following lemmas must be proven first (proof Theorem 1 in [35]). □

Lemma 1 *If the tree of* τ_i *at stage* k *is punished with the possibility of* $c_i(k)$ *and* $c_i(k) = prob[W\tau_i \succ T_k]$ *and* $\lim_{k\to\infty} c_i(k) = c_i^*$, *then for every* $\varepsilon \in (0, 1)$ *and* $k \succ K(\varepsilon)$ *we have* $prob[|c_i^* - c_i(k)| \succ 0] \prec \varepsilon$*(lemma 1 in [35]).*

Proof since the automata used in the proposed algorithm and [35] is the same, we can use the proof of Lemma 1 in [35], and conclude that if c_i represents the final amount of $c_i(k)$ probability, where k is large enough, using the rule of *Weak law of large numbers*, we conclude that $\lim_{k\to\infty} prob[|c_i^* - c_i(k)| \succ \varepsilon] \to 0$. Since for every $\varepsilon \in (0,1)$, there is a $a^*(\varepsilon) \in (0,1)$ and $K(\varepsilon) \prec \infty$, for all $a \prec a^*$ and $k \succ K(\varepsilon)$ we have $prob[|c_i^* - c_i(k)| \succ 0] \prec \varepsilon$, will be completed by proving Lemma 1. □

Lemma 2 *Assume that* $c_j(k) = prob[\overline{W}\tau_j(k+1) \succ T_k]$ *and* $d_j(k) = 1 - c_j(k)$ *are the possibility of penalties and rewards of tree* τ_i *(for all values of* $j = 1, 2, \ldots, r$*) at stage k. If* $\underline{q}(k)$ *is deduced using the proposed algorithm, then the conditional expectation of* $q_i(k)$ *is defined as follows (lemma 2 in [35]):*

$$E[q_i(k+1)|q(k)] = \sum_{j=1}^{r} q_j(k)[c_j(k)q_i(k) + d_j \prod_{e(m,n)\in\tau_i} \delta_n^m(k)] \quad (2)$$

Where r represents all the trees made and

$$\delta_n^m(k) = \begin{cases} p_n^m(k+1) = p_n^m(k) + a(1-\beta_n^m(k))(1-p_n^m(k)); & e_{(m,n)} \in \tau_j \\ p_n^m(k+1) = p_n^m(k) - a(1-\beta_n^m(k))p_n^m(k); & e_{(m,n)} \notin \tau_j \end{cases}$$

Proof Using Lemma 2 in [35] and since the reinforcement scheme used in the proposed algorithm is S-L$_{R-I}$, at any stage of k, the selection probability of tree i, when the selected tree j is punished by the random environment, remains unchanged with the probability of $c_i(k)$ (for all values of $j = 1, 2, \ldots, r$). In other words, when the selected tree j is rewarded, the probability of the edges of the selected tree i that are in tree j are increased by a learning rate and the probability of other edges will be reduced. To show more details on the proof of this lemma, we show it for the Steiner tree graph in Fig. 6. In graph G, the purpose is to find an optimal tree between node 1 and node 5 (Assuming that the data having the requested name by Node 1 is only present at node 5). In this case, the trees available to reach from node 1 to node 5 are as follows [46]. □

We assume that τ_1 is the tree with the lowest weight expected and, $q_i(k)$ is the selection probability of tree τ_i at stage k, so we have:

$$q_1(k) = p_2^1(k)p_4^2(k)p_5^4(k)$$
$$q_2(k) = p_2^1(k)p_5^2(k)$$
$$q_3(k) = p_2^1(k)p_3^2(k)p_5^3(k)$$
$$q_4(k) = p_2^1(k)p_3^2(k)p_4^3(k)p_5^4(k)$$
$$q_5(k) = p_5^1(k)$$

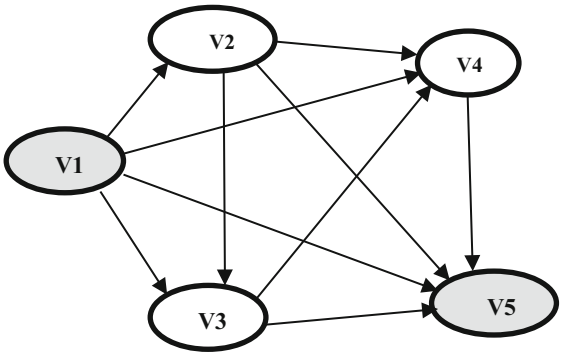


Fig. 6 Graph example to prove the validity of the proposed algorithm

$$q_6(k) = p_4^1(k)p_5^4(k)$$
$$q_7(k) = p_3^1(k)p_5^3(k)$$
$$q_8(k) = p_3^1(k)p_4^3(k)p_5^4(k),$$

$$\begin{aligned}
&E[q_1(k+1)|q(k)] = q_1(k)[c_1q_1(k) + d_1(k)\{p_2^1(k) \\
&\quad + a(1-\beta_1(k))(1-p_2^1(k))\}\{p_4^2(k) + a \\
&\quad (1-\beta_1(k))(1-p_4^2(k))\}\{p_5^4(k) + a(1-\beta_1(k))(1-p_5^4(k))\}] + \\
&q_2(k)[c_2q_1(k) + d_2(k)\{p_2^1(k) + a(1-\beta_2(k))(1-p_2^1(k))\}\{p_4^2(k) - a \\
&\quad (1-\beta_2(k))p_4^2(k)\}\{p_5^4(k) \\
&\quad + a(1-\beta_2(k))(1-p_5^4(k))\}] + \; q_3(k)[c_3q_1(k) + d_3(k)\{p_2^1(k) \\
&\quad + a(1-\beta_3(k))(1-p_2^1(k))\}\{p_4^2(k) - a \\
&\quad (1-\beta_3(k))p_4^2(k)\}\{p_5^4(k) - a(1-\beta_3(k))p_5^4(k)\}] + \\
&q_4(k)[c_4q_1(k) + d_4(k)\{p_2^1(k) + a(1-\beta_4(k))(1-p_2^1(k))\}\{p_4^2(k) - a \\
&\quad (1-\beta_4(k))p_4^2(k)\}\{p_5^4(k) - a(1-\beta_4(k))p_5^4(k)\}] + \\
&q_5(k)[c_5q_1(k) + d_5(k)\{p_2^1(k) - a(1-\beta_5(k))p_2^1(k)\}\{p_4^2(k) - a \\
&\quad (1-\beta_5(k))p_4^2(k)\}\{p_5^4(k) - a(1-\beta_5(k))p_5^4(k)\}] + \\
&q_6(k)[c_6q_1(k) + d_6(k)\{p_2^1(k) - a(1-\beta_6(k))p_2^1(k)\}\{p_4^2(k) - a \\
&\quad (1-\beta_6(k))p_4^2(k)\}\{p_5^4(k) - a(1-\beta_6(k))p_5^4(k)\}] + \\
&q_7(k)[c_7q_1(k) + d_7(k)\{p_2^1(k) - a(1-\beta_7(k))p_2^1(k)\}\{p_4^2(k) - a \\
&\quad (1-\beta_7(k))p_4^2(k)\}\{p_5^4(k) - a(1-\beta_7(k))p_5^4(k)\}] + \\
&q_8(k)[c_8q_1(k) + d_8(k)\{p_2^1(k) - a(1-\beta_8(k))p_2^1(k)\}\{p_4^2(k) - a \\
&\quad (1-\beta_8(k))p_4^2(k)\}\{p_5^4(k) - a(1-\beta_8(k))p_5^4(k)\}]+
\end{aligned}$$

After simplifying all the right-side expressions of the above equation and some algebraic manipulations in Lemma 2 from [35] and Theorem 1 from [46], we achieve the equation presented in Lemma 1, so the proof of the lemma is completed.

Lemma 3 *If* $\underline{q}(k)$ *is edited by the proposed algorithm, conditional expectation* $q_i(k)$ *is always non-negative, i.e.* $\Delta q_i(k) \succ 0$.

Proof According to the definition, we have $\Delta q_i(k) = E[q_i(k+1)|q(k)] - q_i(k)$ and based on Lemma 2, and since the probability of tree creation, based on the edge selection probability for making trees is rewarded or punished, replacing the values of $q_i(k)$ in Eq. (2) and algebraic simplifications, we have $\Delta q_i(k) \succ \prod_{e(m,n)\in\tau_i} \Delta p_n^m(k)$ and the proof of lemma is completed using Lemma 1. □

Proof of Theorem 1 Lemma 3 implicitly indicated that $\{q(k)\}$ is a *Sub-Martingale* [35] and [46]. Using the theory of *Martingale* and the fact that $\{q(k)\}$ is a uniform limit function, we conclude that the amount of $\lim_{k\to\infty} q_i(k)$ converges to q^* with probability of 1. We define the function of ψ as $\psi[\omega, q] = \frac{e^{-\omega q_i/a}-1}{e^{-\omega/a}-1}$ for $\omega \succ 0$. It can be easily shown that this function is *sub-regular*and its value is less than 1 and greater than $1-\varepsilon$. Given that $q_i(k)$is the probability of the construction of minimum Steiner tree $\tau_i$at stage k, and $1-\varepsilon$ is the probability that the proposed algorithm tends to the minimum Steiner tree τ_i, if $\underline{q}(k)$ is edited by the proposed algorithm, then for every error parameter of $\varepsilon \in (0, 1)$, there is a learning rate of $a \in (0, q)$ and $\frac{\omega a}{e^{\omega a}-1} = \max_{j\neq i}(\frac{d_j}{d_i})$ so that $q_i = [q_i(k)|k = 0]$ (it can be easily concluded from the definition of function ψ.) So we deduce that for every error parameter of $\varepsilon \in (0, 1)$ there is a learning rate of $a \in (0, q)$, so that the probability of the proposed algorithm tending to converge to the Steiner tree with the least expected weight is greater than $1-\varepsilon$, therefore the proof of the theorem is done [35].

8 Experimental results

In order to study the performance of the proposed routing algorithm, several simulations have been performed in the NS2 environment. In all these experiments, the simulated environment has1,000 nodes with unique names created using the rules described in the naming section. Each node has 100 named data, as a result, there are 100,000 named data in the entire network and all nodes play both roles of content requester and generator. There are 20 nodes at each site and 10 sites on each network. Sites are the first level of clustering, so 50 clusters were created in the first level. All 10 clusters of the first level were placed in the second level of clustering and the cluster head has information of sub-clusters. A third level is considered for clustering, so that every four clusters in the second level create a higher cluster and the summarized information of the named data are kept in the cluster head. For the cache in the simulation, it is assumed that the cache is only available in cluster head nodes and nodes of each cluster lack cache. One of the main issues discussed at NDN is field matching. In simulations conducted, full matching has been considered and the route is answered if there is a complete matching [8].

Nodes are connected by links with a maximum transmission rate of 2 Mbps. Propagation delays in each link are random value lower than 20 which are obtained by the normal distribution function. A $S-L_{RI}$ learning automata is placed on the first level cluster heads whose number of application is equal to the number of adjacent cluster heads. According to the results of various simulations, learning rate adjustment of 0.02 in the automata gives the best answer and it is 0.2 in the diagrams.

It was observed that 1,000 to 10,000 random requests are presented for the named content among all data available in the network. Simulation time was 250 s and all simulations were repeated 15 times; however, simulation results were measured in 1, 2, 3 and 4 Mbps using CBR current. The paths between the requesters and providers of data are found by LCRN algorithm. After matching field in the content holders, they replied by 2 KB data package and the reverse path was inserted in this package. Specifications of the requested data are added in the cache of intermediate cluster heads. So after a while, the data were not only available in the beginning nodes, but some middle cluster heads also have the data. Action probability vector of automata is updated using the shortest path obtained at each stage and the shortest path cost. According to Dijkstra'salgorithm, and as shown in simulations, the sum of each vector has been always one and the selection probability of the path with the lowest cost is converged to one.

The main unit of measurement for the evaluation of LCRN algorithms in our tests is the average throughput. By definition, it includes the average number of packages that successfully reached the destination per unit of time. Throughput is depicted in Fig. 5 with four different traffics from 1 to 4 Mbps under 1,000–10,000 requests for the named data.

As Fig. 7 shows, when the number of requests for named content increases, the proposed algorithm shows better throughput. The reason for this efficiency and productivity increase is the nature of NDN networks and generally, the content centric networks. Since data is distributed in the network after a while and it is not only in the primary servers, it is easier to be found. On the other hand, algorithm efficiency is increased due to the usage of the learning machine. Figure 8 shows the package delivery rate as a function of the number of requests. This implies that as the throughput increases, the package delivery rate also increases.

In Figs. 9 and 10, successful interest packages rate and the average number of interest packages sent is shown. These figures also reflect the fact that a lot of efforts and errors are initially needed due to the presence of data only in certain servers and also lack of automata knowledge about the environment, but as it is shown in the figures, after the automata obtains estimations about the location of named data, successful interest packages rate gets close to 1 and the number of forwarded packages remains at a minimum.

Figure 11 shows an end-to-end delay in different requests and under different traffics. By increasing the amount of traffic, end-to-end delay increases somehow. When forwarding traffic reaches 4 Mbps, end-to-end delay is not significantly increased. In higher requests, delay had no significant increase because of proper automata learning.

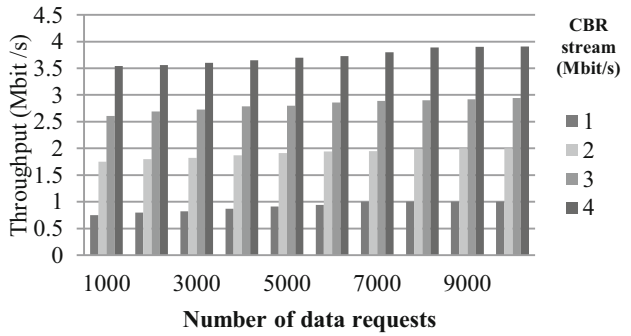


Fig. 7 Average throughput

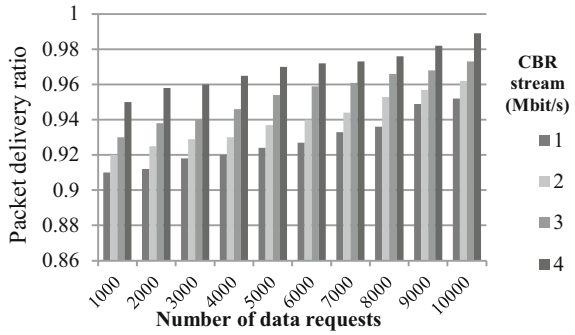


Fig. 8 Packet delivery ratio

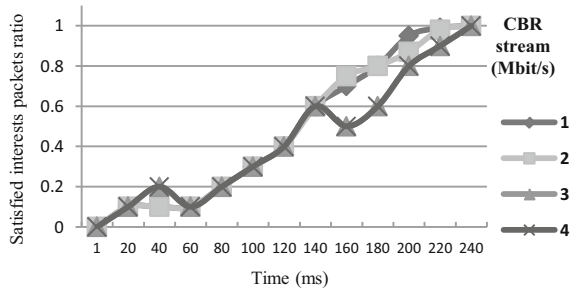


Fig. 9 Satisfied interest package ratio

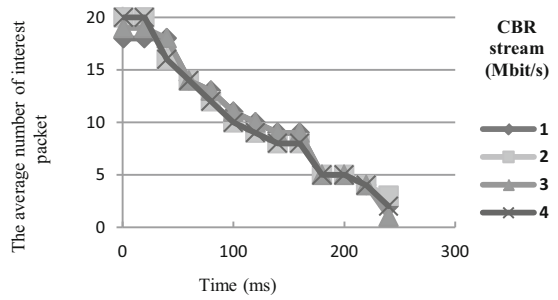


Fig. 10 Average number of interest packages

Figures 12 and 13 show the path creation delay as well as the transmission delay. Simulated conditions are as previously described. By increasing the traffic rate, the amount of both delays increases. However, due to equality in traffics

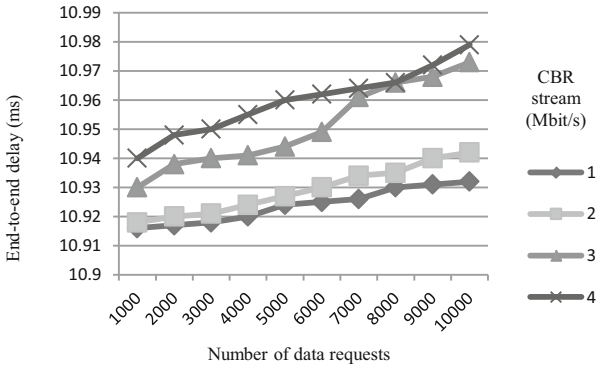


Fig. 11 End to end delay

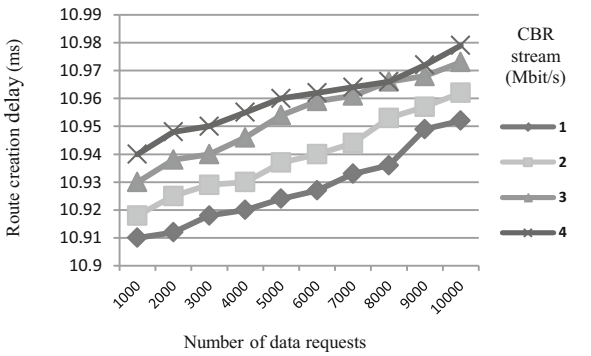


Fig. 12 Route creation delay

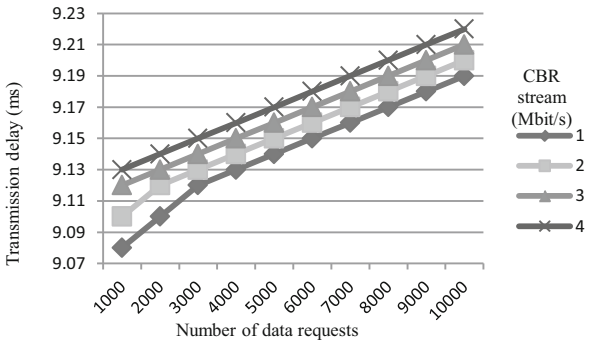


Fig. 13 Transmission delay

and increased number of named data requests, no significant amount has been added to delays.

An important factor in measuring the performance of the network is the package loss rate. In the proposed protocol, package loss rate is the ratio of the forwarded interest packages to the data packages received. This rate is shown in Fig. 14. For all tested traffics (from 1 to 4 Mbps), the amounts of this rate are almost identical. But from observation, when the data request increases, due to data replication in caches of cluster head nodes and greater speed in finding suitable routes, we have reduction in package loss rate and it gets almost close to zero.

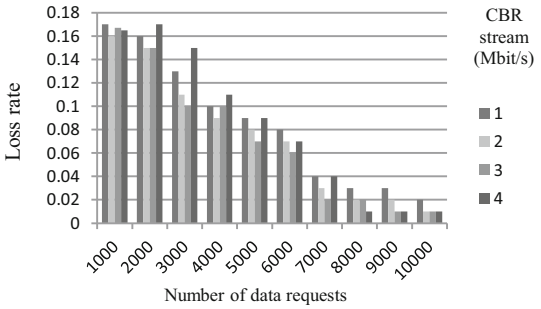


Fig. 14 Packet loss rate of interest

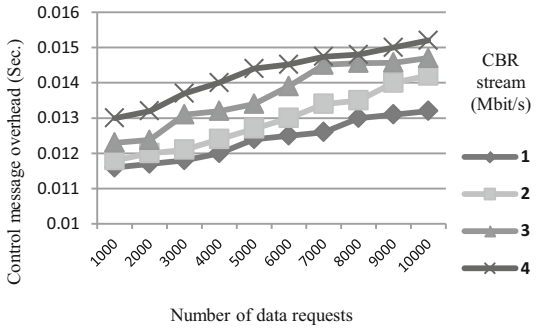


Fig. 15 Control messages overhead to find route

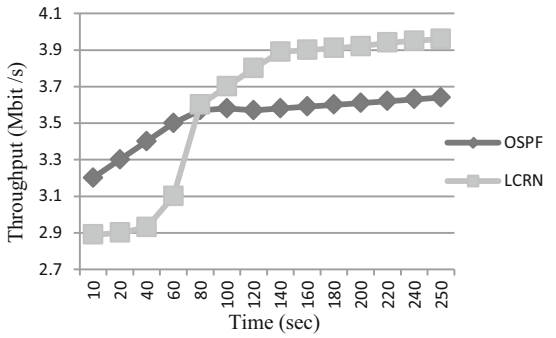


Fig. 16 Throughput LCRN and OSPF

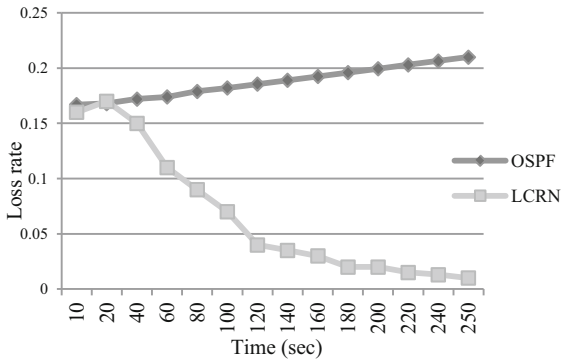


Fig. 17 Loss rate in LCRN and OSPF

Control messages overhead criterion is defined as the number of messages required to find the optimum route forwarded per second. As shown in Fig. 15, this criterion is increased slightly by increase of the number of requests and also increase of network traffic.

In simulation procedure, in order to examine the relative performance of the proposed protocol, some parameters are compared with the routing protocol to open shortest path first (OSPF) [47]. OSPF is a link state routing protocol that can manage the traffics of IP protocols. The reason for this selection is to compare the similarity available in OSPF and LCRN work base.

OSPF routing protocol uses Shortest path first or OSPF algorithm designed by Dijkstra to prevent the routing loop in network topology and creates a loop free network. OSPF has a fast convergence process and also provides incremental update using Link state advertisement (LSA). OSPF is a classless protocol that allows the easy use of Variable length subnet masking (VLSM) and Route summarization for creation of a hierarchical network structure. It should be noted that OSPF is the new basis of OSPFN which is the protocol in progress in NDN [47].

For conducting the above-said comparison, simulation environment is considered as the same with a CBR (Constant bit rate) traffic rate of 4 Mbit/s. Simulations are conducted in 250 s and each simulation is repeated 15 times after which the average parameters are obtained. The average throughput of LCRN and OSPF protocols have been compared in Fig. 16.

As shown in the early stages of implementation, LCRN has lower efficiency than OSPF because head clusters automata lack knowledge about the operational environment, but after a short time that this knowledge is gained, productivity grows.

Figure 17 shows the package loss percentage for LCRN and OSPF protocols. As can be seen, in LCRN, with the passage of time, this value reaches zero, but in OSPF this parameter is ascending. The main reason for this difference is path learning by automata and also data replication in NDN. Since in IP-based networks, routing is always done to get to the destination and no repetition happens, with the passage of time and buffer prolongation, queue overflow and package loss increased too.

Figure 18 shows end-to-end delay for LCRN and OSPF. As expected, this parameter has ascending value of OSPF protocol and it is reversed for LCRN. The reason for this difference in behavior is the same issue that was described for Fig. 17

9 Complexity

Various algorithms may be designed to solve a problem. The best algorithm should be chosen as a measure to compare the efficiency of the algorithm. One of the best measurements is run time complexity. This section describes the computation

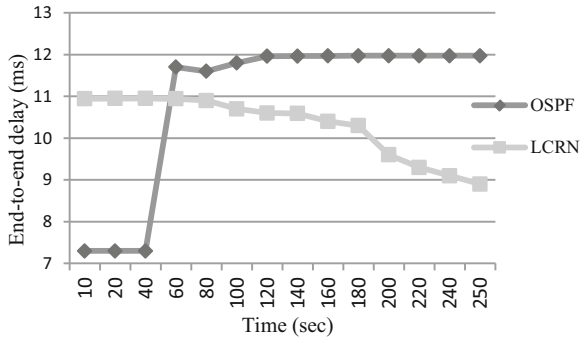


Fig. 18 End to end delay in LCRN and OSPF

complexity of the proposed algorithm, after which a comparison of LCRN with OSPF is done on the basis of complexity. Some of the parameters used in the computation complexity of the algorithm is as follows:

- $G = (V, E)$: Network graph, the V nodes and E edges.
- $n : n = |V|$ Number of nodes in the graph
- $m : m = |E|$ Number of edges in the graph
- k: Number of cluster heads the first level

Lemma 4 *The complexity of LCRN algorithm to calculate the shortest path of NDN, is equal to* $O(k^2)$.

Proof LCRN algorithm in graph G, helps to find the optimal route calls to LR procedure. In Fig. 3, *repeat-until-loop* in the LR procedure, lines 8 to 16, is the main part of the algorithm. The *repeat-until-loop* is executed for each content request. Worst case in the procedure, when checked once all cluster heads. When these conditions occur that are not converging existing automata. As a result, the time complexity of the LR procedure will get $O(k)$. After finding a Steiner tree by the LR procedure, LCRN algorithm acts to reward or penalize automata; for this purpose, it was used from Dijkstra's algorithm (Lines 15 to 25 from LCRN algorithm in Fig. 2) [51,52]. In graph G, if used from Fibonacci heap, Dijkstra's algorithm time complexity is equal to $O(m + n \log n)$. But the algorithm LCRN, due to the use of clustering, is not required to review all network nodes, and only their cluster heads examined. The worst case for this algorithm is $m = k(k-1)/2$, that is happening when all cluster heads are connected together. Thus, the complexity of finding the path by Dijkstra in LCRN is equal to $O(k^2 + k \log k)$. The time complexity of the LCRN algorithm is $O((k)+(k^2+k \log k))$, after simplification changes the $O(k^2)$ and the proof of Lemma 4 ends. □

9.1 Comparison of the complexity of LCRN and OSPF

Since the basis of routing in OSPF is the Dijkstra's algorithm, thus its complexity in the graph $G = (V, E)$ in the worst case is $O(n^2)$, that happens in the full graph [51,52]. Since the $k \succ 0$ is defined as $n \succ k$, consequently it has $O(k^2) \prec O(n^2)$. From this discussion, we conclude that the time complexity of the proposed algorithm is less than that of OPSF.

10 Conclusion

In this paper, a new routing protocol has been provided based on a multi-layer clustering and learning automata in NDNs called LCRN. A routing protocol in NDN was mapped to solve the problem of Steiner tree, so that the content requester and content providers are tree root and terminal nodes, respectively. For the convergence of automata to an appropriate and acceptable answer, Dijkstra's algorithm is used to respond to the environment. If the algorithm finds a path with costs higher than Dijkstra's algorithm, the selected activities are punished and otherwise rewarded.

The proposed idea is simulated using the NS2 simulator. In each simulation, traffic rate of 1–4 Mbps is considered, and also one of the most important NDN network standards which is the request for named data is examined in the range of 1,000–10,000 requests. According to the results and mathematical proof obtained, the algorithm is converging to the optimal Steiner tree. As expected, optimal results cannot be obtained in the beginning of the simulation, but after a while in the simulation, and set learning automata, the algorithm works well and shows good performance. LCRN protocol was compared to the final version of OSPF protocol which is available in NS2, and using graphs, it was shown that the proposed protocol works better.

References

1. Named data networking. http://www.named-data.net/.
2. Project CCNx. http://www.ccnx.org.
3. Drira, W., & Filali, F. (2014). A Pub/Sub extension to NDN for efficient data collection and dissemination in V2X networks. A World of Wireless, Mobile and Multimedia Networks (WoWMoM). In *IEEE 15th International Symposium*, pp. 1–7.
4. Amadeo, M., Campolo, C., & Molinaro, A. (2015). Forwarding strategies in named data wireless ad hoc networks: Design and evaluation. *Journal of Network and Computer Applications*, *50*, 148–158.
5. Mauri, G., Verticale, G. (2013). Distributing Key Revocation Status in Named data networking. *Advances in Communication Networking*, Springer, pp. 310–313.
6. Conti, M., Gasti, P., & Teoli, M. (2013). A lightweight mechanism for detection of cache pollution attacks in named data networking. *Computer Networks*, *57*, 3178–3191.
7. Wählisch, M., Schmidt, T. C., & Vahlenkamp, M. (2013). Backscatter from the data plane-threats to stability and security in information-centric network infrastructure. *Computer Networks*, *57*, 3192–3206.
8. NDN Project. NDN Platform. (2013). http://named-data.net/codebase/platform/.

9. Arianfar, S., Sarolahti, P., Ott, J. (2013). Deadline-based resource management for information-centric networks. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking*, pp. 49–54.
10. Hoque, A., Amin, S. O., Alyyan, A., Zhang, B., Zhang, L., Wang, L. (2013). Nisr: named-data link state routing protocol. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking*, pp. 15–20.
11. Carzaniga, A., Rutherford, M. J., Wolf, A. L. (2004). A routing scheme for content-based networking. INFOCOM 2004. In *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, pp. 918–928.
12. Bari, M., Chowdhury, S., Ahmed, R., Boutaba, R., & Mathieu, B. (2012). A survey of naming and routing in information-centric networks. *IEEE Communications Magazine*, *50*, 44–53.
13. Torres, J., Ferraz, L., & Duarte, O. (2012). Controller-based routing scheme for Named Data Network. Electrical Engineering Program, COPPE/UFRJ, Tech: Rep.
14. Wang, L., Hoque, A., Yi, C., Alyyan, A., & Zhang, B. (2012). OSPFN: An OSPF based routing protocol for named data networking. University of Memphis and University of Arizona, Tech Rep.
15. Dai, H., Lu, J., Wang, Y., Liu, B. (2012). A two-layer intra-domain routing scheme for Named data networking. *IEEE Global Communications Conference (GLOBECOM)*, 2012, pp. 2815–2820.
16. Carzaniga, A., Rutherford, M. J., Wolf, A. L. (2004). A routing scheme for content-based networking. In *Proc. of IEEE INFOCOM*.
17. DiBenedetto, S., Gasti, P., Tsudik, G., Uzun, E. (2011). ANDaNA: Anonymous named data networking application. arXiv:1112.2205.
18. Yi, C., Afanasyev, A., Wang, L., Zhang, B., & Zhang, L. (2012). Adaptive forwarding in Named data networking. *ACM SIGCOMM Computer Communication Review*, *42*, 62–67.
19. Nguyen, A. D., Sénac, P., Ramiro, V., Diaz, M. (2011). Pervasive intelligent routing in content centric delay tolerant networks. In *IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pp. 178–185.
20. Kim, Y., An, J., Lee, Y. -H. (2012). CCNFRR: Fast one-hop Re-Route in CCN. In *IEEE International Conference on Communications (ICC)*, 2012, pp. 5799–5803.
21. Kim, J.-J., Ryu, M.- W., Cha, S.- H., Cho, K. -H. (2013). A cluster based multi-path routing protocol for support load-balancing in content-centric network. In *International Conference on Information Science and Applications (ICISA)*, 2013, 1–2.
22. Eymann, J., Timm-Giel, A. (2013). Multipath transmission in content centric networking using a probabilistic ant-routing mechanism. Mobile Networks and Management, ed: Springer, pp. 45–56.
23. Benoit, A., Brenner, L., Fernandes, P., & Plateau, B. (2004). Aggregation of stochastic automata networks with replicas. *Linear Algebra and its Applications*, *386*, 111–136.
24. Galata, A., Johnson, N., & Hogg, D. (2001). Learning variable-length Markov models of behavior. *Computer Vision and Image Understanding*, *81*, 398–413.
25. Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, *17*, 107–145.
26. Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*, 264–323.
27. Anderson, E., Patterson, D. A. (1997). Extensible, scalable monitoring for clusters of computers. *LISA*, pp. 9–16.
28. Li, C., Liu, W., Wang, L., Li, M., & Okamura, K. (2015). Energy-efficient quality of service aware forwarding scheme for Content-Centric Networking. *Journal of Network and Computer Applications*, *58*, 241–254.
29. Tin-Yu, W., Lee, W.-T., Duan, C.-Y., & Yu-Wei, W. (2014). Data lifetime enhancement for improving QoS in NDN. *Procedia Computer Science*, *32*, 69–76.
30. Bradai, A., Ahmed, T., Boutaba, R., & Ahmed, R. (2014). Efficient content delivery scheme for layered video streaming in large-scale networks. *Journal of Network and Computer Applications*, *45*, 1–14.
31. Yuemei, X., Li, Y., Lin, T., Wang, Z., Niu, W., Tang, H., et al. (2014). A novel cache size optimization scheme based on manifold learning in content centric networking. *Journal of Network and Computer Applications*, *37*, 273–281.
32. Eum, S., Nakauchi, K., Murata, M., Shoji, Y., & Nishinaga, N. (2013). Potential based routing as a secondary best-effort routing for Information-centric networkinging (ICN). *Computer Networks*, *57*(16), 3154–3164.
33. Akbari Torkestani, J., & Meybodi, M. R. (2010). Mobility-based multicast routing algorithm for wireless mobile ad-hoc networks: A learning automata approach. *Computer Communications*, *33*, 721–735.
34. Torkestani, J. A., & Meybodi, M. R. (2012). A learning automata-based heuristic algorithm for solving the minimum spanning tree problem in stochastic graphs. *The Journal of Supercomputing*, *59*, 1035–1054.
35. Akbari Torkestani, J., & Meybodi, M. R. (2011). Learning automata-based algorithms for solving stochastic minimum spanning tree problem. *Applied Soft Computing*, *11*, 4064–4077.
36. Akbari Torkestani, J., & Meybodi, M. R. (2010). Learning automata-based algorithms for finding minimum weakly connected dominating set in stochastic graphs. *International Journal of Uncertainty Fuzziness and Knowledge-based Systems*, *18*, 721–758.
37. Rodolakis, G., Laouiti, A., Jacquet, P., & Meraihi Naimi, A. (2008). Multicast overlay spanning trees in ad hoc networks: Capacity bounds, protocol design and performance evaluation. *Computer Communications*, *31*, 1400–1412.
38. Akbari Torkestani, J., & Meybodi, M. R. (2011). A link stability-based multicast routing protocol for wireless mobile ad hoc networks. *Journal of Network and Computer Applications*, *34*, 1429–1440.
39. Akbari Torkestani, J., & Meybodi, M. R. (2011). A cellular learning automata-based algorithm for solving the vertex coloring problem. *Expert Systems with Applications*, *38*, 9237–9247.
40. Hutson, K. R., & Shier, D. R. (2006). Minimum spanning trees in networks with varying edge weights. *Annals of Operations Research*, *146*, 3–18.
41. Rezaei, Z., & Torkestani, J. A. (2012). An energy-efficient MCDS-based routing algorithm for wireless sensor networks: Learning automata approach. *Przeglad Elektrotechniczny*, *11*, 147–151.
42. Misra, S., Krishna, P. V., Saritha, V., Agarwal, H., & Ahuja, A. (2014). Learning automata-based multi-constrained fault-tolerance approach for effective energy management in smart grid communication network. *Journal of Network and Computer Applications*, *44*, 212–219.
43. Rezvanian, A., Rahmati, M., & Meybodi, M. R. (2014). Sampling from complex networks using distributed learning automata. *Physica A*, *396*, 224–234.
44. Kumar, N., Chilamkurti, N., & Rodrigues, J. J. (2014). Learning automata-based opportunistic data aggregation and forwarding scheme for alert generation in vehicular ad hoc networks. *Computer Communications*, *39*, 22–32.
45. Kumar, N., Tyagi, S., & Deng, D. J. (2014). LA-EEHSC: Learning automata based energy efficient heterogeneous selective clustering for wireless sensor networks. *Journal of Network and Computer Applications*, *46*, 264–279.
46. Beigy, H., & Meybodi, M. R. (2006). Utilizing distributed learning automata to solve stochastic shortest path problems. *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, *14*, 591–615.

47. Lai, W. K., Tsai, C.-D., & Shieh, C.-S. (2008). Dynamic appointment of ABR for the OSPF routing protocol. *Computer Communications*, *31*(14), 3098–3102.
48. Lian, J., Agnew, G. B., Naik, S. (2003). A variable degree based clustering algorithm for networks. In *Proceedings of the 12th International Conference on Computer Communications and Networks, ICCCN 2003*, pp. 465–470.
49. Zhang, H., Liu, H., Jiang, C., & Chu, X. (2015). A practical semi-dynamic clustering scheme using affinity propagation in cooperative picocells. *IEEE Transactions on Vehicular Technology*, *64*(9), 4372–4377.
50. Sourlas, V., Psaras, I., Saino, L., & Pavlou, G. (2016). Efficient hash-routing and domain clustering techniques for information-centric networks. *Computer Networks*, *103*, 67–83.
51. Fortz, B., Thorup, M. (2000). Internet traffic engineering by optimizing OSPF weights. In *IEEE INFOCOM*, pp. 519–528.
52. Giroire, F., Perennes, S., & Tahiri, I. (2013). On the hardness of equal shortest path routing. *Electronic Notes in Discrete Mathematics*, *41*, 439–446.

Zeinab Shariat is a Ph.D. student in the Department of Computer Engineering at Science and Research Branch at Islamic Azad University in Tehran, Iran and has been on the Roudehen Branch at Islamic Azad University faculty since 2011. She received his B.S. degree in Computer Engineering from the Qazvin Branch at Islamic Azad University in 2001, and M.S. degree in Computer Engineering from the Sharif University in 2004 and Ph.D. degrees in Computer Engineering from the Science and Research Branch at Islamic Azad University in 2016. His research interests include wireless networks, roueing and caching in networks, network security, distributed systems and content centric networking.

Ali Movaghar is a Professor in the Department of Computer Engineering at Sharif University of Technology in Tehran, Iran and has been on the Sharif faculty since 1993. He received his B.S. degree in Electrical Engineering from the University of Tehran in 1977, and M.S. and Ph.D. degrees in Computer, Information, and Control Engineering from the University of Michigan, Ann Arbor, in 1979 and 1985, respectively. He visited the Institut National de Recherche en Informatique et en Automatique in Paris, France and the Department of Electrical Engineering and Computer Science at the University of California, Irvine in 1984 and 2011, respectively, worked at AT&T Information Systems in Naperville, IL in 1985–1986, and taught at the University of Michigan, Ann Arbor in 1987–1989. His research interests include performance/dependability modeling and formal verification of wireless networks, distributed real-time systems and cyber-physical systems. He is a senior member of the IEEE and the ACM.

Mehdi Hoseinzadeh is a Assistant Professor in the Department of Computer Engineering at Science and Research Branch at Islamic Azad University in Tehran, Iran. He received his B.S. degree in Computer Hardware Engineering from the Islamic Azad University, Dezful Branch, Dezful, Iran in 2003, and M.S. degree in Computer Engineering from the Department of Electrical and Computer Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran in 2005 and Ph.D. degrees in Computer Engineering from the Department of Electrical and Computer Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran in 2008. His research interests include Social network (design, management, security, analysis), E-marketing & electronic management of market, E-Commerce, E-employment, Network security, Bioinformatics, Radio-frequency identification.