Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A novel forwarding and routing mechanism design in SDN-based NDN architecture*

Jia LI, Ren-chao XIE†‡, Tao HUANG, Li SUN

State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China

†E-mail: renchao_xie@bupt.edu.cn

Received Oct. 25, 2017; Revision accepted Apr. 30, 2018; Crosschecked Sept. 10, 2018

Abstract: Combining named data networking (NDN) and software-defined networking (SDN) has been considered as an important trend and attracted a lot of attention in recent years. Although much work has been carried out on the integration of NDN and SDN, the forwarding mechanism to solve the inherent problems caused by the flooding scheme and discard of interest packets in traditional NDN is not well considered. To fill this gap, by taking advantage of SDN, we design a novel forwarding mechanism in NDN architecture with distributed controllers, where routing decisions are made globally. Then we show how the forwarding mechanism is operated for interest and data packets. In addition, we propose a novel routing algorithm considering quality of service (QoS) applied in the proposed forwarding mechanism and carried out in controllers. We take both resource consumption and network load balancing into consideration and introduce a genetic algorithm (GA) to solve the QoS constrained routing problem using global network information. Simulation results are presented to demonstrate the performance of the proposed routing scheme.

Key words: SDN-based NDN; Forwarding mechanism; QoS routing; Genetic algorithm
https://doi.org/10.1631/FITEE.1700698 **CLC number:** TP393

1 Introduction

Current Internet based on the TCP/IP protocol was originally designed to address the problem of resource sharing, where the basic requirements are connecting hosts in the network and forwarding data packets among a limited number of stationary machines (Ahlgren et al., 2011; Xylomenos et al., 2014). However, with an explosively increasing amount of multimedia services in the network, Internet users are taking more interest in information itself, irrespective of its physical location. Evolved to focus on content distribution and retrieval, the Internet has been dominantly used as a distribution network. As it is complex and error-prone to solve the distribution problem in a host-centric network, along with the inherent problems in an IP network such as security and exhausted naming space, information-centric networking (ICN) has attracted much attention as a novel clean-slate network architecture scheme. Researchers intend to design a network architecture that will replace the current host-centric model.

In the ICN field, named data networking (NDN) has been considered as the most important potential technique. The core principles of NDN are name-based routing and content caching, which can not only enable a network to mark and map contents by their names instead of addresses of hosts, but also shorten the distance between contents and users

‡ Corresponding author
* Project supported by the Fundamental Research Funds for the Central Universities, China (No. 2018PTB-00-03) and the National Natural Science Foundation of China (No. 61501042)
ORCID: Jia LI, http://orcid.org/0000-0002-2022-1674

thanks to available content replicas caches located nearby (Jacobson et al., 2009a). Communication in NDN is driven by data consumers sending interest packets, which in the original forwarding strategy are flooded in the network until reaching the nodes with corresponding content. Because of the apparent flaws of the original flooding scheme, several forwarding strategies have been developed, of which the most frequently used is random unicast. In this strategy the content will be packed into data packets, forwarded in reverse to consumers. Each NDN router maintains three data structures, content store (CS), pending interest table (PIT), and forwarding information base (FIB), which will be searched in turn to forward packets during communication (Jacobson et al., 2009b; Zhang et al., 2014).

As another hot axis of study on network architecture, software-define networking (SDN) is changing the way of designing and managing the network by decoupling the network control plane from the data forwarding plane. The SDN paradigm has been proved to dramatically improve network, resource utilization, simplify management of the network, and promote the continuous evolution of network architecture in a more flexible way (Akyildiz et al., 2014; Feamster et al., 2014).

Recently, research on the integration of NDN and SDN has attracted wide interest for the following reasons: on one hand, SDN provides NDN with a global view of the network and supports centralized control over distributed network entities, benefitting from the fact that the awareness feature of NDN can be fully enhanced (Dixit et al., 2013); on the other hand, NDN is immature and will evolve in multiple ways in the future. The programmable SDN controller enables evolution without a concern for the shackling of infrastructure. The introduction of an SDN controller to the network makes it possible to unicast both interest and data packets in a more optimized path compared with the random unicast scheme in traditional NDN. This can help reduce network traffic and improve the efficiency in terms of delivering data packets (Eum et al., 2015).

There have been some conceptual outlines on the combination of ICN and SDN. Othman and Okamura (2010), Sakurauchi et al. (2010), and Chanda and Westphal (2013) modified the TCP header with a content identifier, so that the SDN switchers could detect the information in the header of packets and direct the packets accordingly. The principle of these approaches is a form of overlay to realize ICN functions in the SDN network. Eum et al. (2015) proposed an ICN architecture within the framework of SDN, taking advantages of SDN and focusing on the combination of two kinds of architectures with the least modification of both architectures. A generalized discussion on naming, caching, and routing issues was presented. Aubry et al. (2015) designed a clean-slate approach using NDN messages and the SDN paradigm, and showed better performance in reducing redundancy and caching compared with traditional architectures. Salsano et al. (2013) discussed an ICN architecture using SDN concepts. They considered mainly the extension of the OpenFlow protocol and proved the effectiveness of utilization of the SDN controller in CONET. Syrivelis et al. (2012) designed an outline of possible realization of ICN solutions along with a novel node architecture, and provided a discussion on the benefits and drawbacks of the node design. Son et al. (2016) proposed a novel forwarding mechanism based on distributed SDN controllers and modification on packet structures. A global forwarding table was used in the controller instead of PIT and FIB in the original NDN.

Although there has been a lot of research on the integration of ICN and SDN, there still exist some problems to be solved. First of all, the flooding scheme adopted in the original NDN will bring about a mess of duplication of packets, which may not only result in poor effectiveness in delivery of data, but also aggravate network traffic overhead and increase resource consumption, especially in a large-scale network. Although the random unicast strategy can address the problem caused by the original flooding strategy, the randomness of the chosen face may lead to a bad solution. On the other hand, traditional FIBs are updated by means of flooding link-state packets, and the delay caused by information flooding may make the chosen face unavailable and thus the request fail. In addition, in the original NDN, routers check FIBs after receiving interest packets. As long as there is no match in the FIBs, the routers abandon the interest packets. Actually, with the limited scope of deployment of NDN, the requested content may be located in other areas of the NDN or even other types of network, like an IP network. It is obviously unwise to discard in-

terest packets immediately without further search. Benefitting from controller global management, the SDN-based NDN architecture can help solve the problems above, while a specific forwarding mechanism still needs to be expounded. Furthermore, in existing work, more attention is paid to the macro design of architecture, and routing strategies carried out in controllers based on quality of service (QoS) are less considered. However, it is quite important to provide QoS-guaranteed services, especially for multimedia applications in the network.

In this paper, we design a novel forwarding mechanism in NDN architecture with distributed controllers, where routing decisions are made globally to solve the problems caused by the random unicast strategy and a scheme of discard. Furthermore, we propose a novel routing algorithm considering QoS applied in the proposed forwarding mechanism and carried out in controllers. We consider both resource consumption and network load balancing and introduce a genetic algorithm (GA) to solve the QoS constrained routing problem using global network information. The main contributions of this study can be summarized as follows:

1. We make some extensions to the general SDN-based NDN architecture. We design the data structures in the controller and router, so that the controller can maintain global information of the local and adjacent network. In addition, the router can record the configuration information from the controller.

2. We propose a novel forwarding mechanism based on SDN-based NDN architecture. Interest packets can be forwarded through an effective centralized routing strategy rather than being flooded and can all be matched, to avoid needless abandonment and thus guarantee a higher response rate.

3. We propose a QoS multi-constrained routing algorithm applied in the proposed forwarding mechanism with consideration of both resource consumption and network load balancing based on an improved GA algorithm, which is carried out in the controllers to make routing decisions during packet forwarding.

2 Forwarding mechanism

In this section, we propose a novel forwarding mechanism in SDN-based NDN architecture, based on which the nodes in the network dispose interest packets and data packets.

2.1 System architecture

We consider an SDN-based NDN architecture. In the architecture, centralized controllers make more efficient and stable management of the network because the controllers maintain the whole image of the network and hold complete global information of all the hosts. Therefore, the controllers in SDN can balance the resource and load of the network so as to make global decisions accurately in time. However, the inherent drawback is that the workload of controllers has a tight relationship with the scale of the network. As the number of nodes and the number of requests in the network increase, the controllers will face massive calculations, and a huge storage system is required to achieve an efficient management. Due to limited computing and storage resources of controllers, and as a single controller in an SDN is vulnerable to a single failure, where flooded requests can saturate a controller's capacity (Wang et al., 2016), it is essential to employ a domain-split technique to spread and balance the workload of controllers.

Some issues need to be considered before embarking on the design of a new network system. First, there is no identification of location in a name-based NDN, where naming and routing are totally based on the name of the content. Therefore, the border nodes that link different domains cannot be addressed by other entities through the IP address used in the current network. In the traditional scheme of NDN, there is no node awareness of the location of a border node except for those connected to it directly, and inter-domain communication is unrealizable. Moreover, in traditional NDN, packet discard is applied as long as a mismatch of forwarding information in local tables occurs. Nodes in the network obtain the registration and location information of content within the network by means of flooding link-state packets within the scope of the local network, so that each node maintains complete information of the network after flooding. In this case, mismatch of interest packets in local tables represents the non-existence of corresponding content within the local network, and the packets will be abandoned immediately. This strategy is definitely impractical in the domain-split situation because the absent content in the local do-

main may exist in adjacent domains or even other kinds of networks. For these reasons, the architecture should realize not only the interaction among controllers to obtain the necessary information of other networks, but also the selection and positioning of border nodes. Furthermore, when an interest packet of the NDN is to be sent to other networks with different protocols like an IP network as representative, or a data packet from an IP network is to be sent back to an NDN network, a gateway needs to be set between the NDN and IP networks. The role of the gateway is to perform protocol conversion and destination redirection.

The instance of architecture to which we refer is presented in the network topology in Fig. 1. The whole NDN network is divided into multiple domains, each of which is autonomous with one local controller (CTR) in the control plane, a number of content routers (CRs), and several border routers (BRs) in the data plane, through which the autonomous domain is connected to others. In addition, there is a content gateway (CGW) between the NDN and IP networks to address protocol conversion. The controllers in the upper plane can carry out interaction with those in adjacent domains.

To design the forwarding mechanism, we make some extensions to this architecture. The controller (CTR) in the control plane maintains the map of the distribution of content in the local network and the content existence table of adjacent networks. The controller includes a routing module which is responsible for locating the position of content according to the path request from the router and calculating the proper path to the content object.

The named-based routing is carried out based

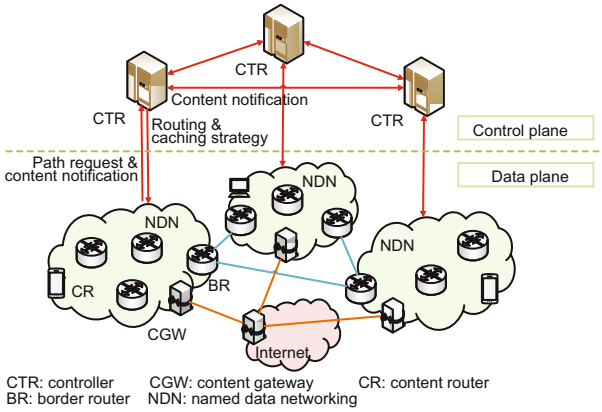

Fig. 1 System architecture

on a routing algorithm such as the Dijkstra algorithm in OSPF or PBR by the routing module in the control plane. The routing algorithm we use is a genetic algorithm. We will explain it in detail later.

To address the position of content and make a path plan, we use a local content forwarding table (LCFT) as shown in Fig. 2 and an adjacent content location table (ACLT) as shown in Fig. 3 to store the routing information. The two fields in LCFT indicate the names of contents that exist in the local network and the IDs of routers where the contents lie. We give the content named A stored in routers 2 and 3 as an example (Fig. 2). In Fig. 3, the fields of ACLT represent the names of the contents, the indexes of domains where the contents are stored, and the IDs of border routers through which the request packets can be sent to the domains. For instance, content C is stored in domain 2 and the corresponding border router is router 3.

Name	Router ID		
A	2	3	···
B	1	4	···
···	···	···	···

Fig. 2 Local content forwarding table structure

Name	Cluster ID	Border router ID
C	2	3
···	···	···
**	Internet	5

Fig. 3 Adjacent content location table structure

There are three cases to be discussed. First, given a global view of the local network topology and the locations of data objects in the LCFT, the routing module can calculate an optimal routing path to the content node. Second, given the existing information of content in other domains in the ACLT, the routing module can choose the proper border node and calculate an optimal routing path to it. Otherwise, the routing module routes the packets to the border nodes connected to the Internet.

In the data plane, there are two kinds of routers: common content routers (CRs) and border routers (BRs). Common content routers are responsible for transferring and caching data. Border routers are the connection points with other domains, so content

registration and security management are carried out here, as well as data transmission and caching.

Each router maintains three structures to forward and cache data: CS, PIT, and FIB. CS maintains the information of content objects stored in the router. PIT records the request information of certain content and the arriving faces, so that the responsive data packets can go back to the request node in the reverse path. FIB is similar to the routing table recording useful routing instruments.

In particular, for the design of the controller, local FIB needs only to maintain partial essential information and will be updated based on times of request. The faces with high value of metric have high probability to forward packets. Their structures are designed as shown in Figs. 4–6.

After the controller calculates a path from the request node to the node where content exists, the whole path information will be sent back to the request node, so that the interest packet can carry the forwarding information with no need to search the tables until arriving at the destination. We design the interest packet structure as shown in Fig. 7, adding the path information and flag field to indicate the existence of available information.

2.2 Forwarding mechanism

At the start of communication, the customer makes a request for certain content using its name in the form of an interest packet. Then the interest packet will be handled step by step progressively as follows:

Name	Data
A	···
B	···
···	···

Fig. 4 Structure of content store

Name	Requesting face	
A	1	3
B	3	···
···	···	···

Fig. 5 Structure of the pending interest table

Name				
	Forwarding face	1	2	···
	Metric	M1	M2	···
	Times of request	5		

Fig. 6 Structure of the forwarding information table

Path flag	Path information	Original CCN interest packet

Fig. 7 Proposed interest packet structure

1. Once the interest packet arrives at a router, the router will look up the PIT first. If there is a match of the content name in the PIT, indicating that there have been other interest packets arriving previously, the arriving face will be added to the request face list of the corresponding content entry. Otherwise, the PIT will build a new item of the content name and add the arriving face to the PIT. Then, go to step 2.

2. The router will look up the CS. If there is a match of the content name, then the data will be packed into a data packet and backtrack through the faces recorded in the PIT, after which the items of the content will be deleted and the interest packet will be discarded. Then, go to step 8; otherwise, go to step 3.

3. The router will check the path flag in the interest packet. If the flag indicates that there exists path information carried by the interest packet, the FIB will be updated based on the path information and the interest packet will be forwarded through the path. Then, go to step 8; otherwise, go to step 4.

4. There is no former path information carried by the interest packet. This means that no path request has been sent to the controller, so the router will look up the FIB first. If there is a match, go to step 5; otherwise, go to step 6.

5. The information in the FIB can be used for forwarding, or a request can be sent to the controller with a certain probability to update the FIB. Therefore, with the given probability, the packet will be sent out through the face with a high metric value in FIB; otherwise, go to step 6.

6. The router will send a path request to the controller. The controller addresses the positions of the routers that store content or the border routers connected to the domains or the CGW to other networks where the content exists as a destination. Then the controller calculates a feasible and optimal path from the request node to the destination node.

7. After the controller gets the path, it sends the whole path information to the request node, which will add it to the head of the interest packet and change the path flag field as well. Then, go back to step 3.

8. If there has been a response to the interest packet, then end. Otherwise, go back to step 1.

The flowchart of the processing of interest packets is presented in Fig. 8.

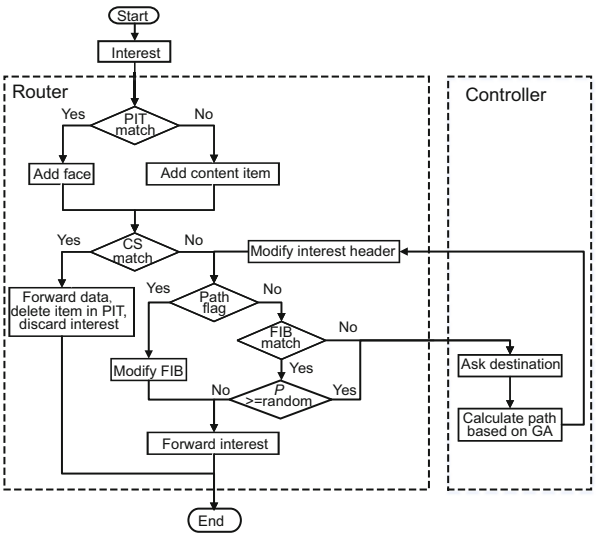

Fig. 8 Flowchart of processing of interest packets

Because the PIT has recorded the coming faces of arriving interest packets, the data packets can perform a backtrace to the source node in the reverse path like a traditional NDN. The data packet will be handled step by step progressively as follows:

1. Once the data packet arrives at a router, the router will look up the content name of the data packet in the PIT first. If there is no match in the PIT, indicating that there has not been any request for the content arriving before and that the data packet is useless, the data packet will be discarded, then end. Otherwise, go to step 2.

2. The router will look up the CS. If there is no match of the content name in the CS, the router will cache the data based on the caching strategy delivered by the controller and send content update notification to the controller so that the controller can update the LCFT.

3. The router will forward the data packet to the faces recorded in the PIT, after which the items of the content will be deleted.

4. If the data packet has arrived at the destination, then end. Otherwise, go back to step 1.

The flowchart of the processing of data packets is presented in Fig. 9.

From the above, we know that controllers play important roles in global network management, including making routing decisions. In addition to the various requirements of multimedia services, the routing algorithm performed in the controllers is significant for optimizing the performance of the network.

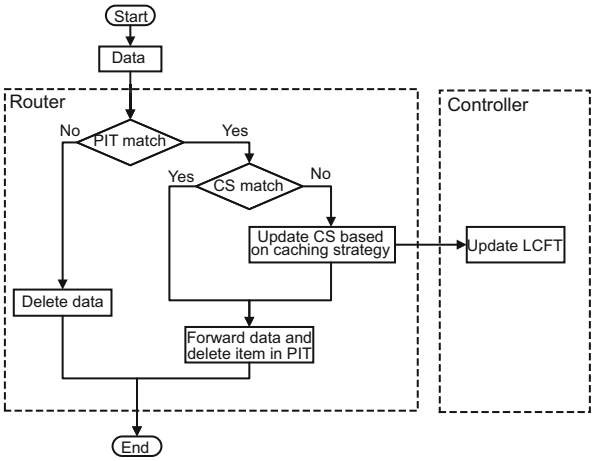

Fig. 9 Flowchart of processing of data packets

3 Routing optimization algorithm

Based on the novel forwarding mechanism designed in SDN-based NDN architecture described in Section 2, we will design a QoS routing algorithm focusing on decreasing resource consumption and balancing network load based on a GA.

3.1 QoS routing problem formulation

The QoS routing problem usually consists of two objectives: (1) finding a path that fulfills the QoS constraints; (2) optimizing the global network performance by finding the best path.

The most primitive and simple routing algorithms are shortest path algorithms in general, such as Yang's strategy (Yang and Sun, 2008) and Wang's strategy (Wang et al., 2014). There has been a lot of research on QoS routing problems in recent years. In many studies, the objective of the routing problem is to find a path that satisfies certain requirements and has the least cost (Shin et al., 2001; Ahn and Ramakrishna, 2002; Zhao and Chen, 2011; Feng et al., 2012; Punhani and Nitin, 2011; Hou et al., 2017). Hou et al. (2008) proposed a strategy similar to those above. Wang and Wang (2001) and Riedl (2002) developed a routing algorithm based on a GA to solve the delay-bandwidth-constraint routing problem, setting the goals of optimization as highest bandwidth and lowest delay. Hou et al. (2008) solved the routing problem with four constraints of delay, bandwidth, loss-rate, and jitter with the use of a GA. As described above, the efficiency of delivery was much focused on; however, the global performance of the network is closely related to the load

balancing such as trade-off between bandwidth utilization and node workload.

We model the network system as a directed graph $G = (V, E)$, where the network nodes are represented by vertices and links are represented by edges. N is the number of nodes in the network. $V = \{v_1, v_2, \ldots, v_N\}$ is the set of network nodes, while E is the set of links in the graph. $e_{ij} \in E$ represents the link between v_i and v_j. A unicast request consists of a source vertex v_s, a destination vertex v_t, and QoS requirements.

In light of different applications served in the network including mainly data, multimedia, and real-time business, four key weights are taken into account, i.e., bandwidth, delay, cost, and load. Here, there are three weights, i.e., bandwidth, delay, and cost, for each link with a delay and load ratio for each node. In this way, for the network, each link can be denoted by a triple tuple $< c, \mathrm{bw}, \mathrm{dl} >$, while each node can be expressed by a binary tuple $< \mathrm{dn}, \mathrm{ld} >$, where c, bw, dl, dn, and ld represent cost, bandwidth, delay of link, delay of node, and load ratio of the node, respectively. The cost of a link in this study is the cost of occupying the link in unit time and it is provided by the network operator.

Based on the above network model, the QoS routing problem with multiple constraints can be described as follows: given network $G = (V, E)$, source node v_s, destination node v_t, and five constraints including bandwidth requirement B, load requirement L, delay requirement D, hop requirement H, and cost requirement C, which are all non-negative real numbers provided by the application or user, the problem is to find a feasible path that can satisfy the above requirements from the source node to the destination node. $p = (v_\mathrm{s}, \ldots, v_i, v_j, \ldots, v_\mathrm{t})$ is one of the paths from v_s to v_t. The metric functions are defined as follows:

$$\rho_{ij} = \begin{cases} 1, & \text{if } e_{ij} \text{ belongs to path } p, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

$$N_k = \begin{cases} 1, & \text{if } v_k \text{ belongs to path } p, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

$$B(p) = \min(\mathrm{bw}_{ij}\rho_{ij}), \quad i, j = 1, 2, \ldots, N, \tag{3}$$

$$L(p) = \max(\mathrm{ld}_k N_k), \quad k = 1, 2, \ldots, N, \tag{4}$$

$$D(p) = \sum_{i=1}^{N}\sum_{j=1}^{N} \mathrm{dl}_{ij}\rho_{ij} + \sum_{k=1}^{N} \mathrm{dn}_k N_k, \tag{5}$$

$$H(p) = \sum_{i=1}^{N}\sum_{j=1}^{N} \rho_{ij}, \tag{6}$$

$$C(p) = \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}\rho_{ij}, \tag{7}$$

where $B(p)$, $D(p)$, $H(p)$, and $C(p)$ represent the bottleneck bandwidth, total delay, number of hops, and total cost of path p, respectively, $L(p)$ represents the highest load ratio of the nodes belonging to path p, bw_{ij}, dl_{ij}, and c_{ij} represent the bandwidth, delay, and cost of the link from v_i to v_j, respectively, and dn_k and ld_k represent the delay and load ratio of v_k, respectively. A selected routing path must meet the requirements below:

1. Available bandwidth constraint: $B(p) \geq B$;
2. Load ratio constraint: $L(p) \leq L$;
3. End-to-end constraint for delay: $D(p) \leq D$;
4. End-to-end constraint for the number of hops: $H(p) \leq H$;
5. End-to-end constraint for cost: $C(p) \leq C$.

We can conclude that in this model there are two bottleneck constraints, available bandwidth constraint and load ratio constraint, of which the values are the minimum or maximum of each link or node in the whole path. The other parameters are all additive, of which the value is the sum of weights of each link and node in the whole path.

Before establishing the objective function, we define several functions.

3.1.1 Resource consumption function $R(p)$

For a selected path p, the resource consumption function can be defined as follows:

$$\begin{aligned} R(p) &= \sum_{i=1}^{N}\sum_{j=1}^{N} B\rho_{ij}D(p) + C(p) \\ &= H(p)BD(p) + C(p), \end{aligned} \tag{8}$$

which is an increasing function of the number of hops, delay, and cost of path p, as well as the required bandwidth. Therefore, the minimum of the resource consumption can imply the objective of the minimum number of hops, delay, and cost given the required bandwidth.

3.1.2 Bandwidth utilization function U_{ij}

The load distribution of a path can be measured by the influence of a new service on the available

bandwidth of links. Therefore, the bandwidth utilization function between nodes v_i and v_j is defined as follows:

$$U_{ij}=\frac{B}{AB_{ij}}. \tag{9}$$

The average and variance of bandwidth utilization are defined as

$$\overline{U}=\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}U_{ij}\rho_{ij}}{H(p)}, \tag{10}$$

$$\delta_{\mathrm{u}}^2=\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(U_{ij}-\overline{U})^2\rho_{ij}}{H(p)}. \tag{11}$$

The variance of bandwidth utilization reflects the load balancing of the whole path.

3.1.3 Node load function

The workload of node v_k is reflected by the load ratio LD_k. The average and variance of the load ratio are defined as

$$\overline{\mathrm{LD}}=\frac{\sum_{k=1}^{N}\mathrm{LD}_k N_k}{H(p)+1}, \tag{12}$$

$$\delta_{\mathrm{l}}^2=\frac{\sum_{k=1}^{N}(\mathrm{LD}_k-\overline{\mathrm{LD}})^2 N_k}{H(p)-1}. \tag{13}$$

The variance of the node load ratio reflects the load balancing of nodes.

3.1.4 Objective function F

Based on the instruments above, our optimization objective is to find a path that satisfies

$$\min\ R(p), \tag{14}$$

$$\min\ \delta_{\mathrm{u}}^2, \tag{15}$$

$$\min\ \delta_{\mathrm{l}}^2, \tag{16}$$

subject to

$$B(p)\geq B, \tag{17}$$

$$L(p)\leq L, \tag{18}$$

$$D(p)\leq D, \tag{19}$$

$$H(p)\leq H, \tag{20}$$

$$C(p)\leq C. \tag{21}$$

Hence, the establishment of the fitness function must satisfy the following conditions:

1. The total resource consumption of the selected route should be minimal.

2. The variance of bandwidth utilization of the selected route should be minimal.

3. The variance of the node load ratio of the selected route should be minimal.

4. The QoS constraints should be satisfied.

According to the above-mentioned conditions, we design the objective function F as

$$\begin{aligned}F=&\alpha R(p)+\beta\delta_{\mathrm{u}}^2+\gamma\delta_{\mathrm{l}}^2+t_1g(z_1)+t_2g(z_2)\\&+t_3g(z_3)+t_4g(z_4)+t_5g(z_5),\end{aligned} \tag{22}$$

where α, β, γ, t_1, t_2, t_3, t_4, and t_5 are positive real coefficients representing the weight of each parameter, indicating the contribution of the corresponding parameter to the objective function. The optimizing characteristic of the objective function is realized by the fact that F decreases with the reduction of resource consumption, variance of bandwidth utilization, and variance of the load ratio, which means that the path with a lower F has better performance and will be more likely to be selected.

The penalty function $g(z_i)$ $(i=1,2,\dots,5)$ is defined to amend the objective function as a punishment on the condition in which the values of the QoS parameters fail to satisfy the constraints.

$$g(z_i)=\begin{cases}0, & \text{if } z_i\leq 0,\\ z_i, & \text{otherwise,}\end{cases}\quad i=1,2,\dots,5, \tag{23}$$

$$z_1=D(p)-D, \tag{24}$$

$$z_2=H(p)-H, \tag{25}$$

$$z_3=C(p)-C, \tag{26}$$

$$z_4=B-B(p), \tag{27}$$

$$z_5=L(p)-L. \tag{28}$$

As mentioned above, a lower value of F represents a better performance of the corresponding path. Therefore, the penalty function $g(z_i)$ is designed to keep the value of the objective function if the constraints are satisfied; otherwise, it will add value to the objective function as a punishment.

3.2 Proposed routing algorithm

It has been proved that the multi-constraint routing problem is NP-hard and difficult to solve. A depth-first search (DFS) algorithm was used in Shin et al. (2001) to find the best path, and a modified particle swarm optimization (PSO) algorithm

was used in Hou et al. (2017) to optimize the routing problem. However, the drawback of these two algorithms is the high possibility of the existence of a local optimal solution. A genetic algorithm (GA) is an effective heuristic algorithm to solve the NP-hard problems described in Goldberg (1989) and Whitley (1994). GA has been widely used in multi-constraint QoS routing. Encoding types fall into two categories: binary (Hou et al., 2008; Feng et al., 2012) and symbol (Wang and Wang, 2001; Ahn and Ramakrishna, 2002; Yussof and See, 2010; Punhani and Nitin, 2011; Zhao and Chen, 2011). Moreover, Yussof and See (2010) provided a general discussion on how to solve the multi-constraint routing problem based on GA. We introduce GA to solve the QoS constrained routing problem in this subsection.

3.2.1 Principle and application on QoS routing of a genetic algorithm

A genetic algorithm is a kind of parallel optimization algorithm that simulates the evolution process of a creature in the method of randomized search, and is suitable for looking for an optimal solution in a large and complicated search space.

The algorithm starts with an initial population which consists of chromosomes in certain numbers. Each solution space is represented by a chromosome and each gene contained in the chromosome is an integral element of the solution space. The forms of encoding the chromosome can be classified as binary and symbolic. By means of a genetic operator of selection, crossover, and mutation, searching for an optimal solution is transferred from one population to a new one of which the fitness value is higher. In the selection process, parent chromosomes with better fitness values are selected to the next generation. In the crossover process, new offspring chromosomes are produced through exchanging a subpart of their parents. To generate new individuals for guaranteeing diversity and pushing the process of optimization, a mutation operator is used to change a little bit of genes with a small probability. The iteration proceeds generation by generation until the convergence conditions are met.

Specifically, when applied to the QoS routing problem, each path from the source node to the destination node can be indicated by one chromosome, containing genes representing nodes in the path.

In the SDN-based architecture proposed in Section 2 and the forwarding strategy of Section 3, the routing algorithm is executed in the controller which maintains an image of the global state of the network beyond the data layer. The distributed searching algorithm will not be practicable for solving the centralized routing problem in the controller. A genetic algorithm is effective in an optimized routing problem with high efficiency based on its global search ability. Therefore, we choose a GA to solve the QoS routing problem calculated in the centralized controller.

3.2.2 Pre-processing of the algorithm

When using the general routing algorithm based on the GA mentioned above, we could obtain the optimal solution by simulation. However, we can further simplify the problem scale according to the inherent nature of the problem. As mentioned earlier, there are two bottleneck constraints: available bandwidth and load ratio. Pre-processing them is conducive to solving the problem more effectively. We can simplify the given topology by deleting those edges whose bandwidths are smaller than the required bandwidth B and pruning the nodes whose load ratios are higher than the specified value L. The new subgraph of the original network definitely satisfies the constraints of bandwidth and load, and the problem is further simplified.

Encoding: In a GA, the most essential issue to solve is encoding, of which the most commonly used two methods are binary and symbolic. For a given routing problem, symbol encoding is the simplest method using the node number for gene encoding. In this way, a sequence of nodes which begins from the source node and ends at the destination node represents a path from the source to the destination. Furthermore, as the solution obtained represents a feasible path, there is no need to carry out an additional decoding process as binary encoding does.

Population initialization: Setting the size of population is a key step in which the efficiency of the algorithm and the cost of memory and time taken by calculation should be balanced. To ensure diversity and randomness of the population, we take a modified depth-first search algorithm in the population initialization process. During the search, the adjacent points of the same node will be sorted at random, so the adjacent nodes will be selected with a random probability.

Fitness function: The design of the fitness function should reflect the optimization objective of the problem. The objective function of the QoS routing problem has been given in Eq. (22).

After the pre-processing of the algorithm, the constraints of bandwidth and load ratio have been satisfied by pruning the nodes and links that do not meet the constraints. Therefore, the objective function can be simplified as

$$F=\alpha R(p)+\beta\delta_{\mathrm{u}}^2+\gamma\delta_{\mathrm{l}}^2+t_1g(z_1)+t_2g(z_2)+t_3g(z_3), \tag{29}$$

$$\begin{aligned}R(p)&=\sum_{i=1}^{N}\sum_{j=1}^{N}B\rho_{ij}D(p)+C(p)\\&=H(p)BD(p)+C(p),\end{aligned} \tag{30}$$

$$\delta_{\mathrm{u}}^2=\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(U_{ij}-\overline{U})^2\rho_{ij}}{H(p)}, \tag{31}$$

$$\delta_{\mathrm{l}}^2=\frac{\sum_{k=1}^{N}(\mathrm{LD}_k-\overline{\mathrm{LD}})^2N_k}{H(p)-1}, \tag{32}$$

$$g(z_i)=\begin{cases}0, & \text{if } z_i\le 0,\\ z_i, & \text{otherwise,}\end{cases}\quad i=1,2,3, \tag{33}$$

$$z_1=D(p)-D, \tag{34}$$

$$z_2=H(p)-H, \tag{35}$$

$$z_3=C(p)-C. \tag{36}$$

According to the definition of GA, the selection of the chromosome is based on the fitness, and a higher value of fitness indicates an individual with better performance. Therefore, we design the fitness function as the reciprocal of the objective function:

$$\text{Fitness}=\frac{1}{F}. \tag{37}$$

3.2.3 Selection

The selection operation is to select chromosomes with higher fitness values from the parent population to obtain the offspring population. We use a roulette wheel method with an elitist strategy to conduct the selection process. In this mechanism, the probability of a chromosome, i.e., a path, being selected to the offspring population is

$$P(p_i)=\frac{\text{Fitness}(p_i)}{\sum_{i=1}^{m}\text{Fitness}(p_i)}, \tag{38}$$

where $\text{Fitness}(p_i)$ is the fitness value of path p_i while $\sum_i^m \text{Fitness}(p_i)$ represents the sum of fitness of all m paths in the population. Every time before the selection, the chromosome with the highest fitness will be copied to the offspring population directly in case the best individual is abandoned during the selection process.

3.2.4 Crossover and mutation

Crossover and mutation ensure that better individuals can be produced and the population can evolve to show better performance. We take one-point crossover and bit mutation in the proposed algorithm.

4 Performance evaluation

In this section, we evaluate the performance of our proposed QoS routing algorithm in a Matlab simulator. We provide the outcome statistics of simulation with different numbers of nodes and essential analysis not only to prove the feasibility of the proposed GA in solving the QoS algorithm, but also to show the better performance of the proposed algorithm compared with other strategies and optimization algorithms at various scales of the network.

The network topological graphs used in the simulation are a kind of modified Waxman network, which uses the k-means clustering method to control the density of nodes in the network graph constructed by Waxman's random graph model. Therefore, the connectivity and homogeneity of the topology will be improved. In the random Waxman graph, the probability of the existence of link l is defined as

$$P_l=\beta\exp(-\frac{d_l^5}{\alpha L}), \tag{39}$$

where α and β control the properties of the network. A greater value of β is more likely to generate a high-density network topology, and a small value of α will increase the proportion of short links to long ones in the generated network. d_l is the distance of the link and L is the border length of the network.

Four strategies are investigated in the simulation: (1) our proposed strategy which considers resource consumption, bandwidth utilization balancing, and node load balancing in path selection; (2) Yang's strategy, designed to find the shortest path with the smallest number of hops; (3) Riedl's strategy, taking delay and bottleneck bandwidth into consideration and aiming at finding a path with low

delay and high bandwidth; (4) Hou's strategy, setting the least cost as the optimal objective, similar to the great majority of mechanisms.

The main parameters in the simulation are presented in Table 1.

Table 1 Simulation parameters

Parameter	Description	Value
Pop_size	Size of population	150
Cross_rate	Probability of crossover	0.9
Mutate_rate	Probability of mutation	0.2
α	Weight of resource consumption	1
β	Weight of bandwidth variance	10 000
γ	Weight of load variance	5000
t_1	Weight of delay punishment	2000
t_2	Weight of hop punishment	8000
t_3	Weight of cost punishment	16 000/7

First, we study the convergence behavior of the proposed GA at different scales of the network with the number of nodes ranging from 15 to 50. Fig. 10 shows the relationship between the iteration number and the best fitness in the population. We use the reciprocal of the best fitness to reflect the characteristic. From Fig. 10, no matter whether the number of nodes increases, the best fitness converges to a certain minimum value as the iterations progress. As the network scale increases, the best fitness also increases.

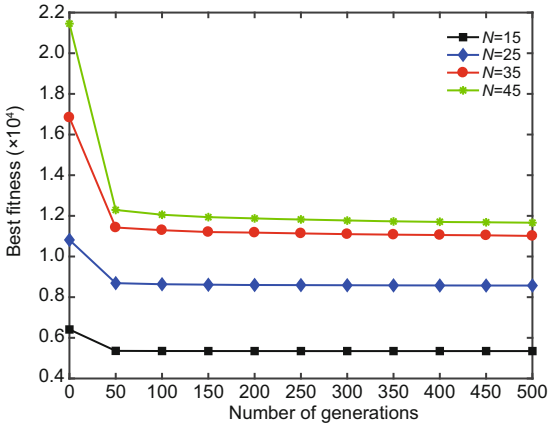

Fig. 10 Convergence of GA when $N = 15, 25, 35, 45$

Fig. 11 compares the performances of the four strategies, among which Riedl's strategy sets delay and bandwidth as the optimal objective, Yang's strategy is aimed to find the shortest path with the minimum number of hops, and Hou's strategy searches for a path with the least cost. We can conclude from Fig. 11 that the proposed algorithm gains the highest fitness and Hou's strategy gets the least. Riedl's strategy performs better than Yang's strategy. We choose the best fitness, number of hops, $(1/B+D)$, and cost as the parameters to be analyzed, where $(1/B+D)$ represents the objective function of Riedl's strategy.

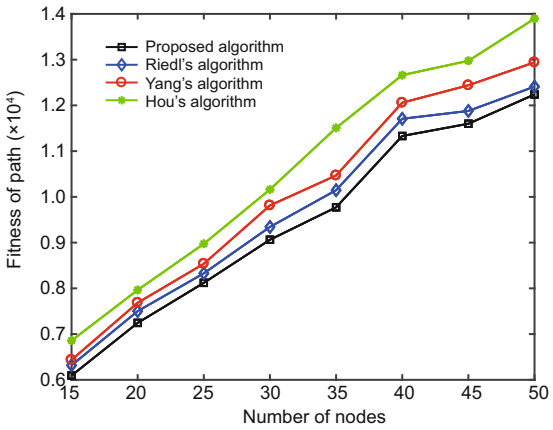

Fig. 11 Fitness of path versus the number of nodes with four algorithms

From Figs. 12–14, we can conclude that compared to the other three strategies, the proposed algorithm does not perform very much worse in terms of the parameters that the other three strategies aim to optimize, while it has much better performance in terms of the best fitness. Furthermore, the proposed routing algorithm can realize lower resource consumption than Yang's and Hou's algorithms (Fig. 15) and better load balancing, i.e., lower variance of load distribution compared to those of the other three algorithms (Figs. 16 and 17).

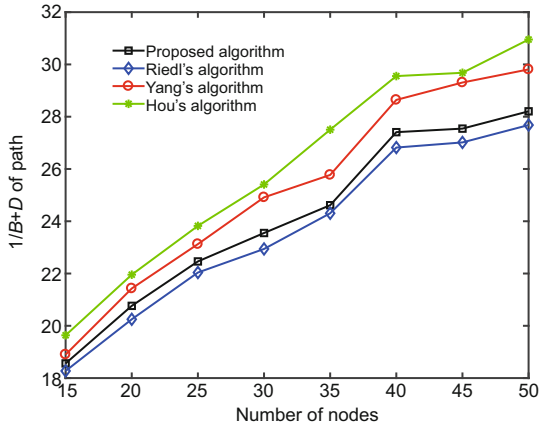

Fig. 12 $1/B + D$ of path versus the number of nodes with four algorithms

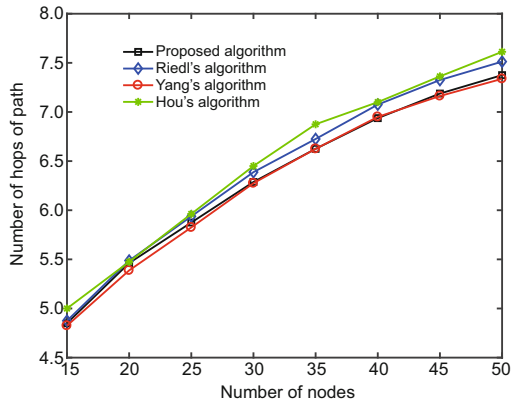

Fig. 13 Number of hops of path versus the number of nodes with four algorithms

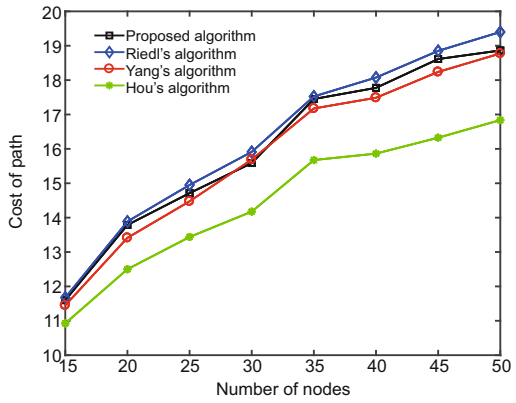

Fig. 14 Cost of path versus the number of nodes with four algorithms

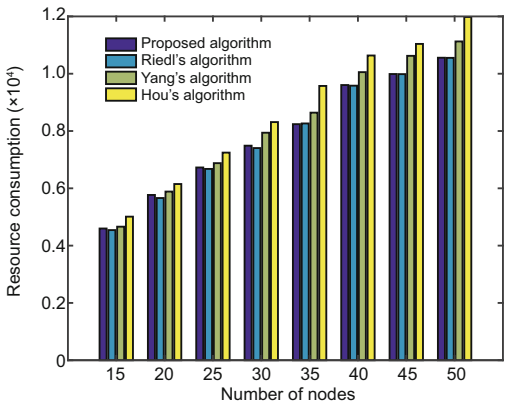

Fig. 15 Resource consumption versus the number of nodes with four algorithms

To analyze the performance of our proposed strategy in specific conditions, we simulate in a deterministic and weighted network topology with 45 nodes (Fig. 18). In the network, each link is denoted by a triple tuple $< c, \mathrm{bw}, \mathrm{dl} >$ marked beside the links, while each node is expressed by a binary tuple $< \mathrm{dn}, \mathrm{ld} >$ marked beside the node, where c, bw, dl, dn, and ld represent the cost, bandwidth, delay of the link, delay of the node, and load ratio of the node, respectively. To demonstrate that the proposed solution can improve network effectiveness in different scenarios, we take into account different positions of content and evaluate the performance separately. In the simulation, node 3 is set as the source and nodes 15, 22, 39, and 40 are set as the destination D, separately.

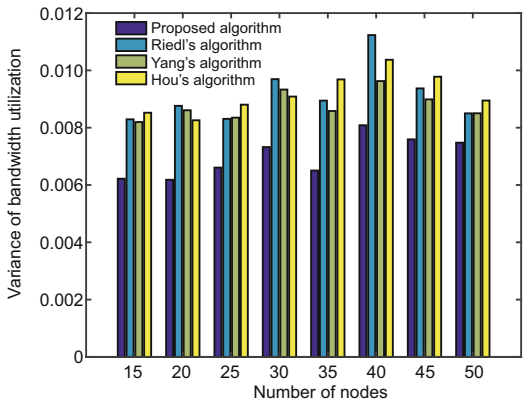

Fig. 16 Variance of bandwidth utilization versus the number of nodes with four algorithms

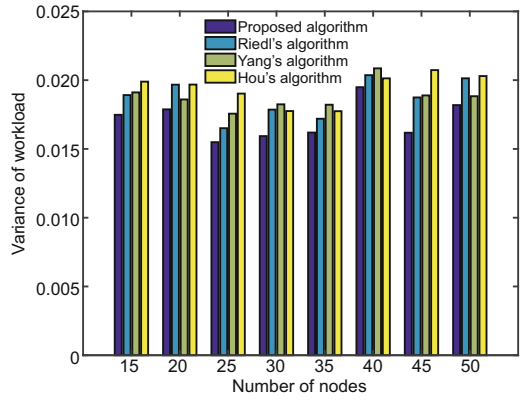

Fig. 17 Variance of workload versus the number of nodes with four algorithms

To facilitate the simulation, we should ascertain the number of iterations of the algorithm. Fig. 10 shows the relationship between the number of iterations and the best fitness in the population, where a convergence is achieved as the iteration number increases to 400 when the number of nodes is 45. Fig. 19 shows the best paths when the destination node ID is 22, found by the four different strategies. Fig. 20 presents the comparison among the four

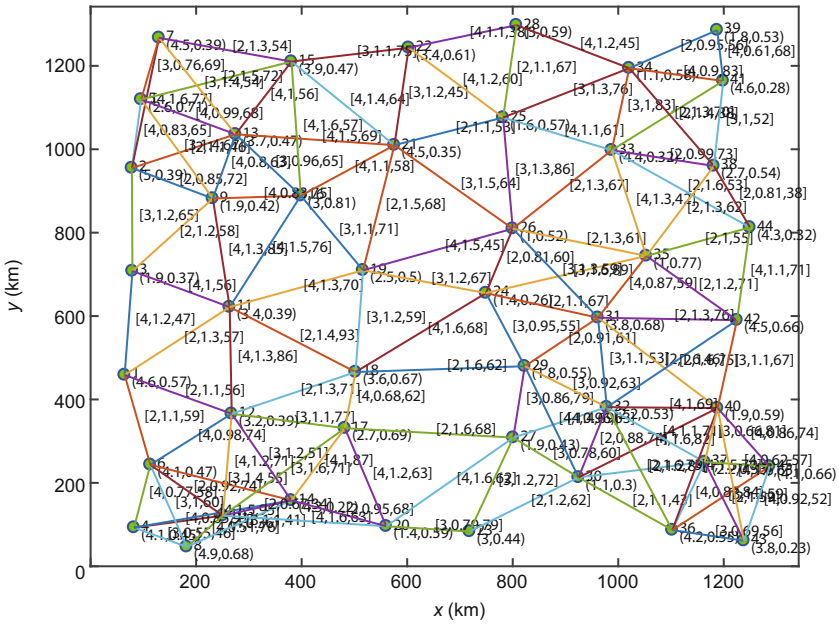

Fig. 18 Network topology of 45 nodes

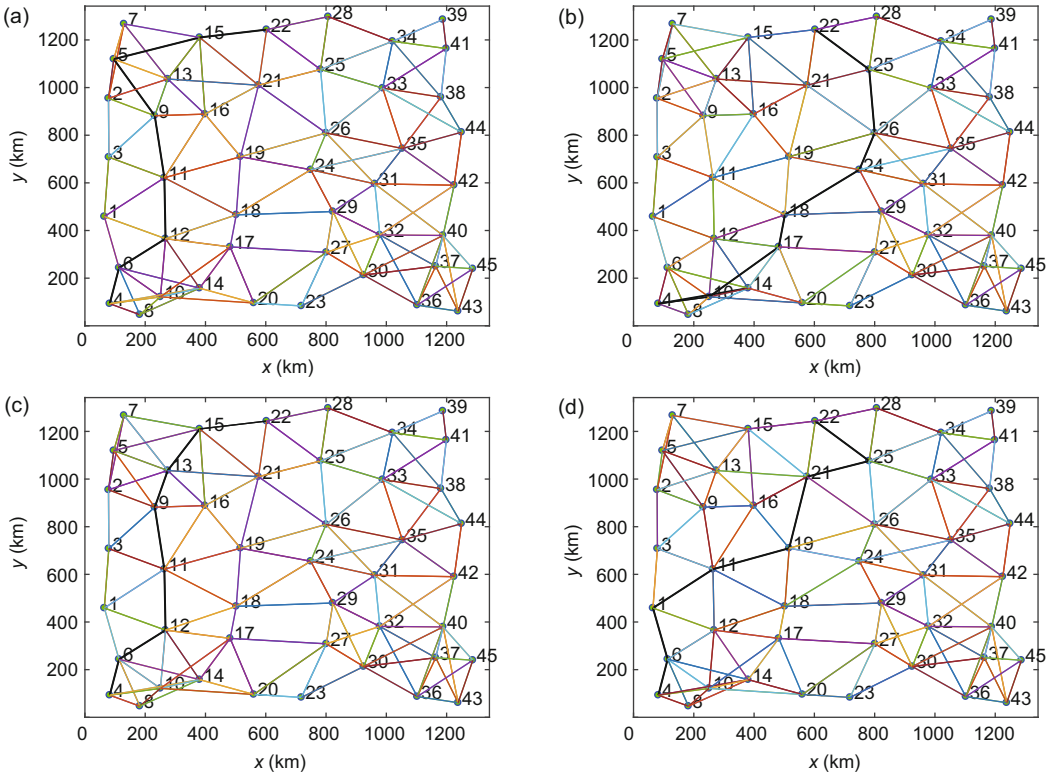

Fig. 19 Path when D=22 calculated by the proposed strategy (a), Riedl's strategy with the highest bandwidth and lowest delay (b), Yang's strategy with the smallest number of hops (c), and Hou's strategy with the least cost (d)

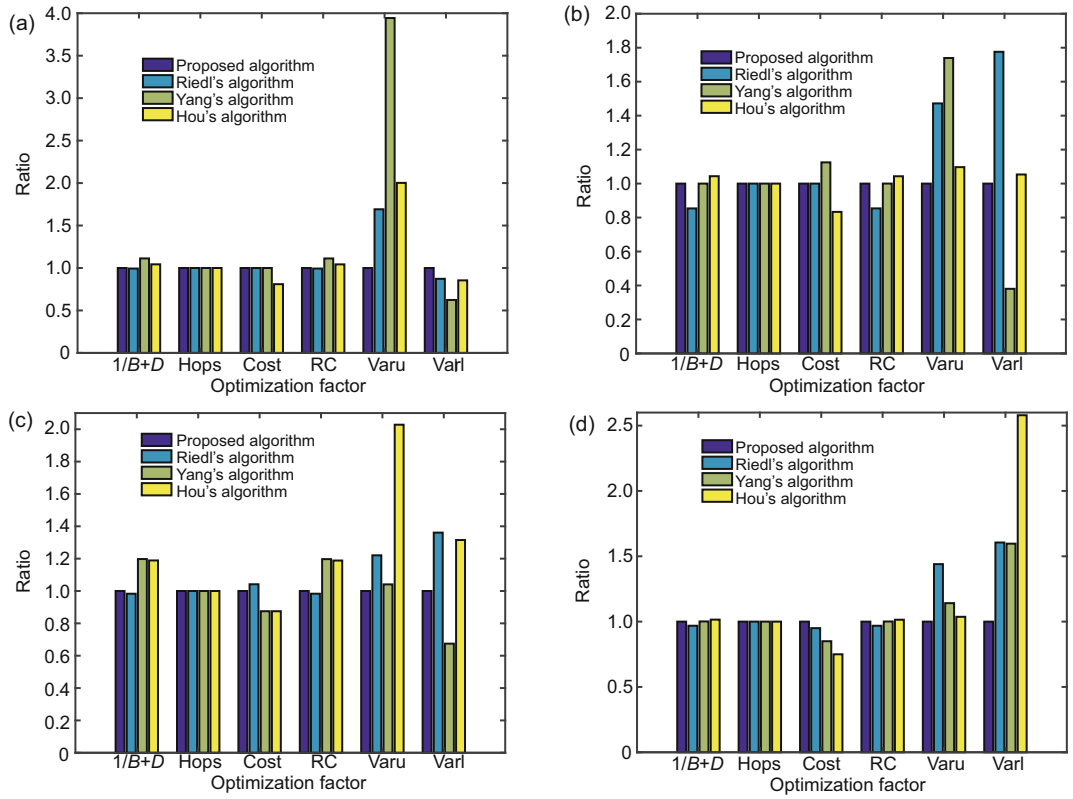

Fig. 20 Comparison of four algorithms in a network of 45 nodes when D=15 (a), D=22 (b), D=39 (c), and D=40 (d)

strategies when D=15, 22, 39, and 40. The comparison of the optimization factors is presented with the result of our proposed strategy as the baseline. Our proposed strategy can always find a path at the cost of performance reduction in terms of one or two factors, but achieves much better performance in the load balancing of the network. As an example, when $D = 22$, although the value of $1/B + D$ in Riedl's strategy is 0.8–1.0 times ours, the values of Varu and Varl are 1.4–1.8 times ours. In addition, the values of Varu and Varl in Yang's strategy and Hou's strategy are too high or unbalanced compared to the proposed strategy, and the other factors are worse than or nearly equal to ours. Therefore, we can conclude that our proposed strategy has a better performance in routing optimization in consideration of resource consumption and load balancing of the network.

To demonstrate that the proposed GA has a better performance in the multi-constraint optimization problem, the other three optimization algorithms are applied to our proposed strategies to find the optimal path. Fig. 21 shows the optimization results of the four algorithms when the number of nodes increases.

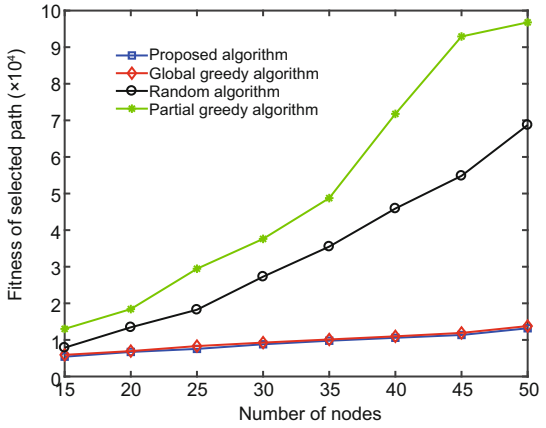

Fig. 21 Fitness of path versus the number of nodes with four algorithms

Compared to other algorithms, GA can always find a better path with the lowest fitness. The first global greedy algorithm has the second best performance just behind GA. The global greedy algorithm has a similar principle to the Dijskra algorithm and can achieve an optimal solution in simple conditions and a near optimal solution to an NP-hard problem such as the multi-constraint routing

problem. The random algorithm has worse performance than the former two algorithms, which can be deduced from the fact that every step of forwarding node selection is totally random, just dislodging the path, which cannot meet the QoS constraints and the solution is obtained without any optimal procedure. The fourth local greedy algorithm has the worst performance because in this algorithm, during each step of node selection, only the performance of the next hop is considered, even though the heading is totally deviated from the right direction to the destination. The situation will become worse when the number of nodes increases, as it is more likely that the destination is farther from the nodes selected by the greedy algorithm at the beginning. The worst situation in the greedy algorithm is that every other link is better than the link connected to the destination node and the solution will traverse all the nodes in the network.

5 Conclusions

In this paper, we propose an extension to the general SDN-based NDN architecture in reference to distributed controllers in the SDN. Based on the proposed architecture, we put forward a novel forwarding mechanism to deal with information redundancy caused by the flooding strategy in traditional NDN networks. Furthermore, the proposed forwarding mechanism realizes non-abandonment of interest packets through routing management in the controller. We propose a global QoS routing algorithm executed in the controller based on the proposed forwarding mechanism, considering both resource consumption and load balancing of the network, and calculated by a modified GA. Simulation results have been presented to show that our proposed QoS routing algorithm can find a path with lower resource consumption and better load balancing. In addition, the GA has better performance than the three other algorithms in solving the QoS routing problem.

References

Ahlgren B, Dannewitz C, Imbrenda C, et al., 2011. A survey of information-centric networking. *IEEE Commun Mag*, 50(7):26-36. https://doi.org/10.1109/MCOM.2012.6231276

Ahn CW, Ramakrishna RS, 2002. A genetic algorithm for shortest path routing problem and the sizing of populations. *IEEE Trans Evol Comput,* 6(6):566-579. https://doi.org/10.1109/TEVC.2002.804323

Akyildiz IF, Lee A, Wang P, et al., 2014. A roadmap for traffic engineering in SDN-openflow networks. *Comput Netw*, 71:1-30. https://doi.org/10.1016/j.comnet.2014.06.002

Aubry E, Silverston T, Chrisment I, 2015. SRSC: SDN-based routing scheme for CCN. Proc IEEE Conf on Network Softwarization, p.1-5. https://doi.org/10.1109/NETSOFT.2015.7116130

Chanda A, Westphal C, 2013. ContentFlow: mapping content to flows in software defined networks. https://arxiv.org/pdf/1302.1493v2.pdf

Dixit A, Hao F, Mukherjee S, et al., 2013. Towards an elastic distributed SDN controller. *ACM SIGCOMM Comput Commun Rev*, 43(4):7-12. https://doi.org/10.1145/2534169.2491193

Eum S, Jibiki M, Murata M, et al., 2015. A design of an ICN architecture within the framework of SDN. Proc 7th Int Conf on Ubiquitous and Future Networks, p.141-146. https://doi.org/10.1109/ICUFN.2015.7182521

Feamster N, Rexford J, Zegura E, 2014. The road to SDN: an intellectual history of programmable networks. *ACM SIGCOMM Comput Commun Rev,* 44(2):87-98. https://doi.org/10.1145/2602204.2602219

Feng J, Jiang N, Wang SQ, 2012. Distributed QoS routing algorithm based on Partheno-GA. Proc Int Conf on Intelligent System Design and Engineering Application, p.247-250. https://doi.org/10.1109/ISdea.2012.444

Goldberg DE, 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Co., Reading, Mass.

Hou HF, Li F, Wang HY, 2008. QoS multicast routing algorithm with multiple constraints based on GA. Proc Int Conf on Machine Learning and Cybernetics, p.1374-1378. https://doi.org/10.1109/ICMLC.2008.4620619

Hou R, Chang YZ, Yang LQ, 2017. Multi-constrained QoS routing based on PSO for named data networking. *IET Commun*, 11(8):1251-1255. https://doi.org/10.1049/iet-com.2016.0783

Jacobson V, Smetters DK, Briggs NH, et al., 2009a. VoCCN: voice-over content-centric networks. Proc Workshop on Re-architecting the Internet, p.1-6. https://doi.org/10.1145/1658978.1658980

Jacobson V, Smetters DK, Thornton JD, et al., 2009b. Networking named content. Proc 5th Int Conf on Emerging Networking Experiments and Technologies, p.1-12. https://doi.org/10.1145/1658939.1658941

Othman OMM, Okamura K, 2010. Design and implementation of application based routing using OpenFlow. Proc 5th Int Conf on Future Internet Technologies, p.60-67. https://doi.org/10.1145/1853079.1853096

Punhani A, Nitin, 2011. A QoS based routing using genetic algorithm. Proc World Congress on Information and Communication Technologies, p.793-797. https://doi.org/10.1109/WICT.2011.6141348

Riedl A, 2002. A hybrid genetic algorithm for routing optimization in IP networks utilizing bandwidth and delay metrics. Proc IEEE Workshop on IP Operations and Management, p.166-170. https://doi.org/10.1109/IPOM.2002.1045774

Sakurauchi Y, McGeer R, Takada H, 2010. Open web: seamless proxy interconnection at the switching layer. Proc Int Conf on Networking and Computing, p.285-289. https://doi.org/10.1109/IC-NC.2010.19

Salsano S, Blefari-Melazzi N, Detti A, et al., 2013. Information centric networking over SDN and OpenFlow: architectural aspects and experiments on the OFELIA testbed. *Comput Netw*, 57(16):3207-3221. https://doi.org/10.1016/j.comnet.2013.07.031

Shin DW, Chong EKP, Siegel HJ, 2001. A multiconstraint QoS routing scheme using the depth-first search method with limited crankbacks. Proc IEEE Workshop on High Performance Switching and Routing, p.385-389. https://doi.org/10.1109/HPSR.2001.923666

Son J, Kim D, Kang HS, et al., 2016. Forwarding strategy on SDN-based content centric network for efficient content delivery. Proc Int Conf on Information Networking, p.220-225. https://doi.org/10.1109/ICOIN.2016.7427118

Syrivelis D, Parisis G, Trossen D, et al., 2012. Pursuing a software defined information-centric network. Proc European Workshop on Software Defined Networking, p.103-108. https://doi.org/10.1109/EWSDN.2012.20

Wang HZ, Zhang P, Xiong L, et al., 2016. A secure and high-performance multi-controller architecture for software-defined networking. *Front Inform Technol Electron Eng*, 17(7):634-646. https://doi.org/10.1631/FITEE.1500321

Wang XH, Wang GX, 2001. An algorithm for QoS routing to optimize network resource utilization. Proc Int Conf on Info-Tech and Info-Net, p.474-479. https://doi.org/10.1109/ICII.2001.983623

Wang Y, Ma XL, Lao YT, et al., 2014. A two-stage heuristic method for vehicle routing problem with split deliveries and pickups. *J Zhejiang Univ-Sci C (Comput & Electron)*, 15(3):200-210. https://doi.org/10.1631/jzus.C1300177

Whitley D, 1994. A genetic algorithm tutorial. *Stat Comput*, 4(2):65-85. https://doi.org/10.1007/BF00175354

Xylomenos G, Ververidis CN, Siris VA, et al., 2014. A survey of information-centric networking research. *IEEE Commun Surv Tutor*, 16(2):1024-1049. https://doi.org/10.1109/SURV.2013.070813.00063

Yang Q, Sun T, 2008. Routing protocol for wireless sensor networks based on least hop. *Comput Eng*, 34(22):129-131 (in Chinese). https://doi.org/10.3969/j.issn.1000-3428.2008.22.044

Yussof S, See OH, 2010. A robust GA-based QoS routing algorithm for solving multi-constrained path problem. *J Comput*, 5(9):1322-1334.

Zhang LX, Afanasyev A, Burke J, et al., 2014. Named data networking. *ACM SIGCOMM Comput Commun Rev*, 44(3):66-73. https://doi.org/10.1145/2656877.2656887

Zhao YF, Chen XY, 2011. Improvement and analysis of GA on finding QoS routing. Proc Int Conf on Electronic & Mechanical Engineering and Information Technology, p.4293-4296. https://doi.org/10.1109/EMEIT.2011.6023990