Journal of Communications and Information Networks, Vol.2, No.3, Sept. 2017
DOI: 10.1007/s41650-017-0028-4
© Posts & Telecom Press and Springer Singapore 2017

Research paper

Special Focus on Future Internet

Fast network restoration with equivalent cooperative routing

Fang Dong, Jianyang Huang, Penghao Sun, Yuxiang Hu*

National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450002, China

* Corresponding author, Email: chxachxa@126.com

Abstract: Aiming at the problem of failure recovery in current networks, a fast failure recovery method based on equivalent cooperative routing is proposed. Firstly, the transmission path between the source and destination nodes is divided into several non-overlapping path segments. Next, backup paths are deployed for each link in the path segment through segmented routing technology, which ensures fast routing recovery after failure. Additionally, in order to avoid damaging the QoS of the data stream through the failure recovery process, the transmission is guaranteed by the intersegment QoS complement. The experimental results show that the proposed method has a low failure recovery delay under a relatively small flow table cost.

Keywords: fast failure recovery, backup path, segment routing, equivalent cooperative routing

Citation: F. Dong, J. Y. Huang, P. H. Sun, et al. Fast network restoration with equivalent cooperative routing [J]. Journal of communications and information networks, 2017, 2(3): 10-17.

1 Introduction

With the rapid development of multimedia technology, the Internet has become the world's largest information application deployment platform. There are many key applications (such as: VoIP, streaming media, gaming, e-commerce, and electronic meetings, etc.) that are highly sensitive to packet loss and delay. Network failures caused by packet loss or long delays will have a great impact on the service quality of these applications. However, the widely deployed routing protocols such as OSPF[1] and IS-IS[2] are required to perform route aggregation and path recalculation after network failure, and then retransmit the lost packets. This kind of solution has a high packet loss rate and a long delay of failure recovery, thus it is difficult to meet critical transmission quality requirements of many services.

Failure recovery has always been a hotspot in the research of communication networks. Depending on whether the alternative path is calculated before a failure occurs, the current failure recovery scheme can be divided into two parts: passive and proactive.

The passive failure recovery scheme recalculates the route from the centralized controller or switch after a failure occurs, and then completes the retransmission of the packet. Refs. [3,4] designed a passive failure recovery scheme for a SDN-based operator network. Each time a failure occurred, the switch notified the controller of the topology change, and then the controller completed the calculation of the failure recovery path and updates of relevant flow table. However, the program produced both a large failure recovery delay and controller overhead. The FCP (Failure-Carrying Packets) scheme

Manuscript received May 28, 2017; accepted Jul. 17, 2017
This work is supported by the National Basic Research Program of China (“973” Program) (No. 2013CB329104).

proposed in Ref. [5] enables all packet headers to carry any encountered failure link information. The intermediate forwarding nodes ignore the failure link and perform the path recalculation after receiving these packets; thusly, they do not need route aggregation and can provide multi-link failure protection. However, this scheme requires each packet to carry failure link information and routing recalculation for each hop in-between, which will bring a larger header storage overhead and transmission delay. Computational overhead and recovery delay are two major shortcomings of passive failure recovery schemes. Although domestic and foreign researchers have proposed different improvement schemes, it is still difficult to completely eliminate such disadvantages.

A proactive failure recovery scheme calculates the backup path for the link before a failure occurs. When the link fails, the failure detection node can immediately switch traffic to the backup path to implement fast route recovery. Traditional proactive failure recovery schemes, such as the KF (Keep Forwarding) scheme proposed in Ref. [6], prioritize the out ports of the router for different destination addresses in the network initialization phase, and every routing node generates d (the number of router ports) different routing entries for each destination address. The router selects an out port according to priority when forwarding a packet. When a link failure occurs, the router can select other ports to forward the packet, which can accomplish fast multi-failure route recovery. However, this scheme increases the necessary storage space of the flow table by d times.

Ref. [7] proposed an Ethernet failure recovery scheme based on segment protection. In this scenario, the backup path of each link was calculated in advance. When a failure occurred, the switch did not need to recalculate the route, and could switch the packet directly to the backup path, which has a small failure recovery delay. Ref. [8] held the view that current SDN mechanisms used by the fault monitoring and recovery model spent too much time on failure discovery and notification. Therefore, in order to reduce the time of failure detection, it extended the OpenFlow1.1 protocol, and had switches in the data plane send periodic probe messages to monitor each flow, thus achieving rapid detection and announcement of failure. Based on OpenState, Ref. [9] proposed a fast failure recovery mechanism of the data plane. This mechanism used the backtracking packet to carry out the failure information announcement. The rerouting node switched the transmission path after the reception of the backtracking packet, thus preventing a routing loop and at the same time reducing the failure recovery delay to a certain extent.

In general, proactive failure recovery has a small recovery delay, but requires a larger support of routing table storage, while passive failure recovery is just the contrary. To achieve rapid failure recovery, this paper proposes an ECRFR (Equivalent Cooperative Routing based Fast Failure Restoration) scheme using proactive recovery. ECRFR divides the transport path between the source and destination node pairs into several non-overlapping segments (as shown in Fig. 1), and the nodes at each end of each segment can be reached from any of the included paths. Under normal circumstances, the shortest path of the segment is chosen for transmission, and others act as backup. When there is a failure in the shortest path, nodes at both ends of the segment can quickly switch the data stream to the backup path to realize rapid route recovery.

Indeed, network architecture based on cooperation is also a promising design, such as the SINET (Smart Collaborative Identifier Network)[10,11], which has already been applied in scalable routing, efficient mapping systems, mobility management, security issues, and so on. We believe this novel network architecture based on cooperation will greatly benefit future Internet design thanks to high degrees of flexibility and security, manageability, mobility support, and efficient resource utilization.

In addition, if the QoS (Quality of Service) of the whole stream cannot be guaranteed by path switching in a particular segment, ECRFR improves the overall QoS by changing the transmission priority of the stream in other segments. As shown in Fig. 1, path P_1 of $Seg2$ is the shortest path, and when a failure takes place in P_1, node1 can quickly switch the data flow to another path. If the transmission delay of the new path is too large, then ECRFR will bring the overall transmission delay within a reasonable range by changing the queue priority of such streams in other segments. ECRFR is a fast and efficient failure recovery scheme to guarantee the transmission quality of data flow by means of proactive failure recovery and equivalent cooperation among fragments.

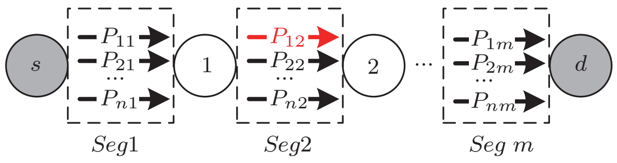


Figure 1 An implemented example of ECRFR

2 Equivalent cooperative routing based fast fault restoration

ECRFR is a way of segmenting the flow paths and complementing the effects between segments to satisfy the overall QoS. As shown in Fig. 1, the capacity of service quality that segments can provide may be different, but the overall guarantee of the quality of the stream can be achieved by the complementary capabilities of different segments.

2.1 Basic definitions

For convenience, and as a detailed description of the ECRFR implement, we define some common terms as follows:

Definition 1 (path segment): ECRFR divides the streaming paths between the source and destination nodes into segments, each of which is called a path segment. All paths contained in the path segments are not necessarily equivalent in hops, but any path can connect to the nodes at both ends of the path segment. For example, $Seg1 \sim Seg\ m$ in Fig. 1 are called path segments, and all paths $P_1 \sim P_{nj}$ $(1 \leqslant j \leqslant m)$ can connect to the nodes at both ends.

Definition 2 (key node): The nodes at both ends of the path segment are called key nodes, and the key nodes near the source node are called the root key nodes; the key nodes near the destination node are called the sink key nodes.

Definition 3 (parallel path): All paths contained in path segment $Seg\ j$ $(1 \leqslant j \leqslant m)$ are called parallel paths. For example, $P_1 \sim P_{nj}$ $(1 \leqslant j \leqslant m)$ in Fig. 1 are called the parallel paths of corresponding path segments.

2.2 Mathematical model

In ECRFR, the acyclic path set among the source and destination node pairs (s, t) is denoted as $P = n_1, \cdots, n_k$ $(k \geqslant 1)$. The path set between adjacent key nodes (n_j, n_{j+1}) is path segment $Segj$, node n_j is the root key node of the path segment, and node n_{j+1} is the sink key node of $Segj$. Due to different transmission capacities of parallel paths in the path segment, the quality of the streaming service provided by different path segments is different. Assuming that the transmission capacity of each parallel path $P_1 \sim P_{nj}$ $(1 \leqslant j \leqslant m)$ in path segment $Segi$ $(1 \leqslant i \leqslant m)$ is respectively $A_1 \sim A_{nj}$ $(1 \leqslant j \leqslant m)$, then the transmission capacity of path segment $Segi$ $(1 \leqslant i \leqslant m)$ is the maximum of these values:

$$A_{Segi} = \max\{A_1, A_2, \cdots, A_{nj}\}.$$

Assuming that the demanded quality of flow f between the source and destination nodes (s, t) is Q_f, and the path segment sequence is $P_1 \sim P_{nj}$ $(1 \leqslant j \leqslant m)$, then to complete the transmission of f, there must be

$$Q_f \leqslant A_{Seg1} + A_{Seg2} + \cdots + A_{Segm}.$$

Assuming that the real-time service quality of path segment $Segi(1 \leqslant i \leqslant m)$ can provide for flow f is Q'_{segi}, then the normal transmission of f should be

$$Q_f = A'_{Seg1} + A'_{Seg2} + \cdots + A'_{Segm}.$$

The QoS provided by each path segment may be different. If the quality of service provided by a path segment is not enough, ECRFR can guarantee the QoS of the data stream by improving the QoS of this stream in other path segments.

For a given source and destination node pair (s, t), a very important step is to determine all of its corresponding path segments. Because the distinction among different path segments mainly relies on key nodes on the two ends, if all the key nodes are marked out, the partition of the path segment will naturally show up. This paper presents a key node marking algorithm, as shown in Algorithm 1. Fig. 2 shows the implementation of Algorithm 1. The processing of this algorithm marks all the key nodes in the left topology (key nodes are marked in dark grey).

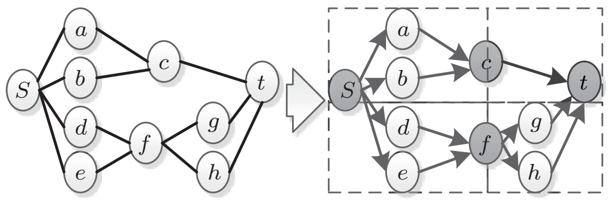


Figure 2 The marking process of key nodes

Algorithm 1 Key node marking algorithm

```
1. Establish a spanning tree with $s$ as the root and $t$ as the leaf, with other nodes arranged in layers according to their minimal hops from $s$;
2. Delete the links between nodes that have a layer gap larger than 1, and keep only links between nodes from the same or adjacent layers;
3. Mark source node $s$ and destination node $t$ as key nodes;
4. Traverse all nodes between $s$ and $t$. If the previous key node of the current node has more than one path with the shortest distance to the current node, mark it as a key node.
```

3 Fast failure recovery based on equivalent cooperative routing

A proactive failure recovery scheme calculates the backup path for the link before any failure occurs. When the link fails, the failure detection node can immediately switch traffic to the backup path to implement fast route recovery. In this paper, a proactive recovery scheme based on equivalent cooperative routing is proposed, which mainly includes two aspects: i) In the same path segment, the parallel path is stored as the backup path of the link at the critical node, and the fast route recovery can be realized after a failure occurs; ii) if the path switching within a path segment affects the service quality, then other path segments are adjusted so that the overall quality of the service flow is satisfied.

3.1 Selection and deployment of backup paths

Multiple parallel paths in the same path segment can be mutually backed up. When a link in a route fails, route recovery can be completed through the link's parallel path. Due to different transmission capacities of each parallel path, in order to make full use of network resources, the parallel path with the strongest transmission capacity has higher priority and the parallel path with lower priority is used as the backup path for the former.

In the same path segment, the backup path of the link is stored in the root key node. In order to reduce storage overhead, this backup path is represented by SR (Segment Routing)[12,13]. SR was proposed by the IETF (Internet Engineering Task Force) in 2013. It is a source routing technology that stores the data forwarding path in the form of a node tag sequence at the source node. Each node tag represents a node, and each intermediate node controls the packet forwarding according to the routing information carried by the packet header. Fig. 3 shows an example of failure recovery in a path segment, where nodes a and h are respectively the root key nodes and the sink key nodes of such path segment. Path $\{a \to c \to e \to g \to h\}$ has the highest priority, so path $\{a \to b \to d \to f \to h\}$ is the backup path for all links. When link $\{c \to e\}$ fails, the root key node a pushes the node tag sequence (h, f, d, b, a) to the packet header. The intermediate nodes then perform a "pop" or "forward" operation on the package according to the top node label. Finally, when all the node tags are processed, the packets are forwarded to key node h.

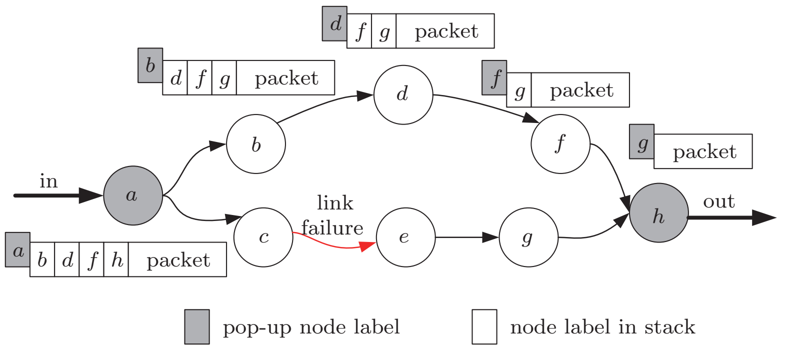


Figure 3 Failure recovery process in a path segment

3.2 QoS complement among path segments

Rapid failure recovery in a certain path segment may not guarantee overall service quality of the stream. By adjusting the other path segments, the service quality complement between path segments satisfies the overall quality of service of the flow. When adjusting other path segments, the origin path should remain unchanged, and raising the packet priority in the queue of switch nodes should be completed first. Combined with the description of the previous section, this paper presents a fast failure recovery algorithm based on ECRFR (as shown in Algorithm 2).

```
Algorithm 2   Fast failure recovery algorithm
1. If the backup path of the failed link is still able to satisfy
   the QoS of data stream $f$, skip to procedure 5;
2. Traverse parallel paths in the path segment. If there is
   a path $P$ that does not need to be adjusted for other
   path segments, the QoS of data stream $f$ can be guaranteed.
   When the backup path is switched to $P$, skip
   to procedure 5;
3. Raise the switch port queue priority of data stream $f$
   in each path segment in turn. If the QoS is guaranteed,
   skip to 5;
4. Adjust the transmission path of data stream $f$ in each
   path segment one by one so that the QoS of each path
   segment $f$ is maximized until the QoS requirement of $f$
   is satisfied;
5. End.
```

4 Simulation

In order to verify the performance of the proposed method (ECRFR) on network failure, this paper compares the performance of FCP, KF, and OpenState on three performance indexes, including recovery delay, storage cost, and resource adjustment ability. In the experiment, we selected a three-network topology (specific parameters shown in Tab. 1).

Table 1 Experiment topology

topology	node amount	link amount	link amount/node amount
Abilene	11	14	1.273
Redlris	18	30	1.667
Cisco	76	160	4.21

In the experiment, a multi-link failure is simulated with a certain frequency and the average duration of failure. The distribution of link failure is the same as the Poisson distribution, with a mean of 3 times every 2 min, and the duration of failure is equal to the exponential distribution with a mean of 30 s. The link bandwidth is set to 1 G and the link delay is 10 ms. The route calculates the optimal path based on the number of hops between nodes.

4.1 Recovery delay

Fig. 4 shows the average failure recovery delay of each algorithm under different network topologies. It can be seen from the data in this figure that since FCP is a passive failure recovery, it is necessary to recalculate a route every time a link failure takes place, thus creating a large recovery delay. In addition, with the expansion of network topology, the probability of link failure in the network is also increased, so the average recovery delay of FCP increases with the expansion of topology. KF, OpenState, and ECRFR in the three topologies have an average recovery delay of 8.9 ms, 10.47 ms, and 8.77 ms, respectively. Because they are proactive, when a failure takes place, the failure detection node can directly forward the affected packets to the preset backup path, so it has a low recovery delay.

4.2 Storage cost

The rapid growth of the size flow table is one critical issue facing constant development of the Internet, so it is necessary to evaluate the additional routing storage cost of different schemes.

Supposing the degree of node n_i is d, all links connected to it are denoted as $l_1, l_2, \cdots, l_d$, and the backup path length corresponding to link l_j $(1 \leqslant j \leqslant d)$ is $D(l_j)$. To represent each node in an MPLS network, only 32 bits are needed, so the flow table storage cost of a link backup on node n_i with segment routing in ECRFR is $32 \times \sum_{j=1}^{d} D(l_j)$ bits. If the average length of backup paths in all links is $AvgP$, then the storage cost increases when ECRFR and OpenState are used in node n_i is

$$C = 32 \times d \times \lceil AvgP \rceil .$$

By this calculation, the average length of backup paths using ECRFR in each of the three topologies is 1.2, 1.64, and 2.2, while those of OpenState are 2.58, 3.64, and 4.85. The average flow table cost of different networks in a traditional network is denoted as S_{avg1}, S_{avg2}, S_{avg3}. Since KF needs at least d times more storage space, and the average port number of a network is $2 \times E/V$, where E and V represent the link number and node number, respectively, the average flow table storage cost of KF in different networks is $2 \times E \times S_{\text{avg1}}/V$, $2 \times E \times S_{\text{avg2}}/V$ and $2 \times E \times S_{\text{avg3}}/V$. Tab. 2 displays the average flow table cost in different failure recovery schemes.

Table 2 Average flow table cost

network	Abilene	Redlris	Cisco
FCP	S_{avg1}	S_{avg2}	S_{avg3}
KF	$2.55 \times S_{\text{avg1}}$	$3.33 \times S_{\text{avg2}}$	$4.21 \times S_{\text{avg3}}$
OpenState	$S_{\text{avg1}} + 106$	$S_{\text{avg2}} + 195$	$S_{\text{avg3}} + 654$
ECRFR	$S_{\text{avg1}} + 49$	$S_{\text{avg2}} + 88$	$S_{\text{avg3}} + 297$

It can be concluded from Tab. 2 that FCP does not increase the flow table storage cost; the storage cost increments of OpenState in the Abilene, Redlris, and Cisco are 106, 195, and 654 bits, respectively; the storage cost increments of ECRFR in these different topologies are 49, 88, and 297 bits respectively, while KF of the three networks brings about 2.55, 3.33, and 4.21 times the storage cost. Compared with FCP, ECRFR has certain increases in flow table storage costs, but these increments are acceptable. Compared with KF, ECRFR saves about d times the storage cost and enhances performance.

4.3 Resource adjustment ability

In order to verify the resource adjustment ability of ECRFR, FCP, KF, and OpenState, it is necessary to count the total number of links in which the load ratio exceeds the set threshold during failure recovery. In this experiment, when selecting the Cisco network as the topology, k $(k = 1, 2, \cdots, 5)$ link failures are generated in the network with random numbers, and the source and destination nodes are also generated. The bandwidth between source and destination nodes is B Mbit/s, and the duration is 1 min. In order to simulate a real network, the user must set background traffic between some nodes at 10 kbit/s, and assure that the number of nodes generating background traffic obeys the Poisson distribution with a mean of 30 s. The total duration of each test is 2 h, and includes counting the total number of links in which the load ratio exceeds the set threshold during failure recovery.

Fig. 5 shows the resource adjustment ability for each scenario under the Cisco topology. The threshold is set to 80%. The experimental results show

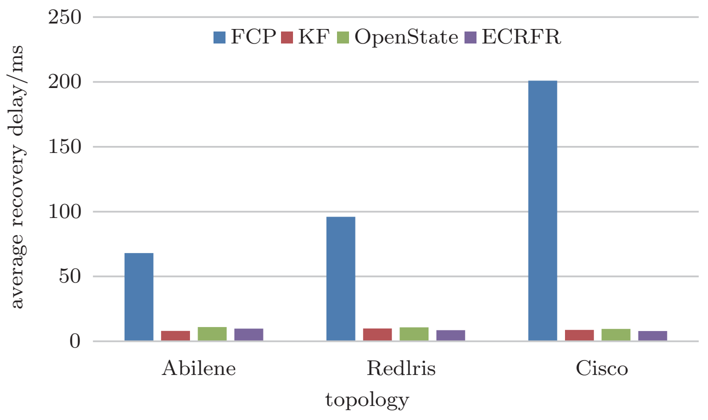


Figure 4 Recovery delay in experiment topology

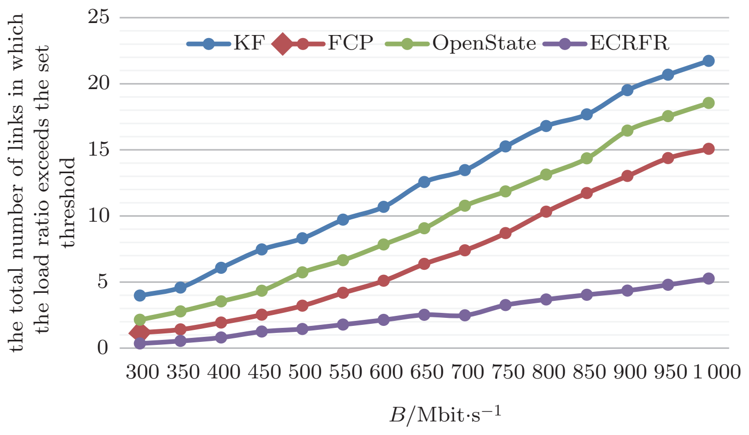


Figure 5 Resource adjustment comparison

that within 2 hours, the total number of links in which the load rate exceeds the threshold in different schemes is 22, 15, 19, and 5, respectively, and the number of ECRFRs is only 22.7%, 33.3%, and 26.3% in the first three schemes. Therefore, in the process of network failure recovery, ECRFR adjusts network resources with a QoS complement among different path segments to avoid the high probability of load ratio. This performance can not only provide better QoS for the data flow, but also promote the rational use of network resources.

5 Conclusion

This paper focuses on the problems of failure recovery in current networks, summarizes the limitations of existing methods, and proposes a fast failure recovery method based on equivalent cooperative routing. Firstly, the transmission path between the source and destination nodes is divided into several non-overlapping path segments, and then the network is protected by proactive failure recovery and the intersegment QoS complement. In order to reduce the flow table cost, the proposed method deploys the backup path for links in the path segment with segment routing. The experimental results show that the proposed scheme has a low failure recovery delay under a relatively small flow table cost. At the same time, this scheme adjusts the resources of each path segment by way of the intersegment QoS complement, which provides the required QoS for data flow.

References

[1] J. Moy. OSPF version 2 [R]. RFC 2328, 1998.

[2] D. Oran. OSI IS-IS intra-domain routing protocol[R]. RFC 1142, 1990.

[3] S. Sharma, D. Staessens, D. Colle, et al. Enabling fast failure recovery in OpenFlow networks [C]//8th International Workshop on the Design of Reliable Communication Networks (DRCN), Krakow, Poland, 2011: 164-171.

[4] D. Staessens, S. Sharma, D. Colle, et al. Software defined networking: meeting carrier grade requirements [C]//IEEE Workshop on Local & Metropolitan Area Networks, Chapel Hill, USA, 2011: 1-6.

[5] K. Lakshminarayanan, M. Caesar, M. Rangan, et al. Achieving convergence-free routing using failure-carrying packets [J]. ACM SIGCOMM computer communication review, 2007, 37(4): 241-252.

[6] B. H. Yang, J. D. Liu, S. Shenker, et al. Keep forwarding: towards k-link failure resilient routing [C]//IEEE INFOCOM, Toronto, Canada, 2014: 1617-1625.

[7] A. Sgambelluri, A. Giorgetti, F. Cugini, et al. OpenFlow-based segment protection in Ethernet networks [J]. IEEE/OSA journal of optical communications & networking, 2013, 5(9): 1066-1075.

[8] J. Kempf, E. Bellagamba, A. Kern, et al. Scalable fault management for OpenFlow[C]//IEEE International Conference on Communications (ICC), Kyoto, Japan, 2011: 6606-6610.

[9] A. Capone, C. Cascone, A. Q. T. Nquyen, et al. Detour planning for fast and reliable failure recovery in SDN with OpenState [C]//2015 11th International Conference on the Design of Reliable Communication Networks (DRCN), Kansas City, USA, 2015: 25-32.

[10] H. Zhang, W. Quan, H. C. Chao, et al. Smart identifier network: a collaborative architecture for the future Internet [J]. IEEE network, 2016, 30(3): 46-51.

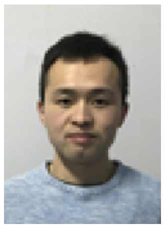
[11] H. Zhang, P. Dong, W. Quan, et al. Promoting efficient communications for high-speed railway using smart collaborative networking [J]. IEEE wireless communications, 2015, 22(6): 92-97.

[12] R. Hartert, S. Vissicchio, P. Schaus, et al. A declarative and expressive approach to control forwarding paths in carrier-grade networks [J]. ACM SIGCOMM computer communication review, 2015, 45(5): 15-28.

[13] F. Hao, M. Kodialam, T. V. Lakshman. Optimizing restoration with segment routing [C]//IEEE INFOCOM 2016, San Francisco, USA, 2016: 1-9.

About the authors

Fang Dong was born in 1981. She received the B.E. degree in communication and information engineering from National Digital Switching System Engineering & Technological R&D Center. Her research interests include novel network architecture, software-defined networks, network evaluation and so on. (Email: chxachxa@163.com)



Jianyang Huang was born in 1992. He received the B.E. degree in communication and information engineering from National Digital Switching System Engineering & Technological R&D Center. His research interests include software-defined networks, novel switching and routing. (Email: 15136143225@163.com)

Penghao Sun was born in 1992. He received the B.E. degree in communication and information engineering from National Digital Switching System Engineering & Technological R&D Center. His research interests include novel network architecture and programmable forwarding (Email: sphshine@126.com)

Yuxiang Hu [corresponding author] was born in 1982. He received the Ph.D. degree in communication and information engineering from National Digital Switching System Engineering & Technological R&D Center. His research interests include future Internet, novel switching and routing. (Email: chxachxa@126.com)