Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 8, August 2019, pp. 2199-2207
DOI: 10.23940/ijpe.19.08.p20.21992207

Fast Multipath Jump Algorithms for Security Constraints

Wanwei Huang[a], Yang Chen[c], Jianwei Zhang[a,*], Chunfeng Du[b], and Sunan Wang[d]

[a]*Software Engineering College, Zhengzhou University of Light Industry, Zhengzhou, 450002, China*
[b]*School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou, 450002, China*
[c]*National Digital Switching System Engineering Technology Research Center, Zhengzhou, 450002, China*
[d]*School of Electronics and Communication Engineering, Shenzhen Polytechnic College, Shenzhen, 518055, China*

Abstract

Transmission path jump can effectively resist network reconnaissance attacks by fragmented data transmission. However, most of the existing path jump models are based on SMT or game theory, which results in an exponential increase of computing time with network size. An efficient path generation algorithm for active random routing is proposed. Firstly, based on the global view of the network defined by the software, the paths of multiple streams are randomly changed actively and concurrently to resist reconnaissance, eavesdropping, and DoS attacks. Secondly, the traditional path calculation algorithm is re-modelled to satisfy the capacity, security, and QoS constraints of the *K* path generated, while improving the computational efficiency. Then, the optimal *K* value is solved, and the security effect of dynamic path is analysed. Finally, simulation results based on typical network topology show that the proposed algorithm can avoid and defend against malicious eavesdropping by attackers and improve the computational efficiency.

Keywords: software-defined networking; multi-path; path jump

(Submitted on June 11, 2019; Revised on July 6, 2019; Accepted on August 7, 2019)

1. Introduction

Traditional single-path routing schemes with minimum cost are powerless in the face of DoS attacks. Such routing simplifies accessibility and manageability, but it provides "useful" help for opponents to gradually learn network routing and accurately plan DoS flooding attacks. For example, an intruder can simply interrupt network data transmission by attacking one of the intermediate nodes along an associated path. Since only one predictable route is selected, the intruder can easily find the route and attack it. The dynamic network can transfer attack surface irregularly [1-2], so that the attacker can neither acquire the information of the defender accurately nor guarantee the accessibility of attack packet transmission. It can effectively defend against node eavesdropping and DoS attacks.

Current intra-domain routing and forwarding strategies in network infrastructure usually adopt dynamic routing protocols such as OSPF and IS-IS [3-5], which can dynamically change the transmission path of traffic according to network topology changes (such as link, node failure, etc.) or QoS policies. From the perspective of dynamic defence, the dynamic transmission of traffic can be improved in traditional dynamic routing systems by adding an "active transformation" strategy. By dynamically changing the transmission paths of the two communicating parties, the malicious interception of the attacker is circumvented and defended, thereby increasing the difficulty and cost of attack detection. However, as a distributed routing protocol, each routing node advertises state changes to neighbouring nodes and then updates to the entire network. If each node changes state too frequently, it will cause a "route convergence" problem. The SDN-based centralized control architecture can effectively avoid this problem [6-7].

Multipathing is widely used in existing networks [8], such as dynamic paths for load balancing or reliability. However, the goal is to achieve improved transmission performance rather than increase the transmission of hidden lines, so it is not possible to initiate eavesdropping or DoS attacks on specific nodes or links in the link. In addition, the existing random path jump algorithms are mostly based on SMT or game theory. The calculation time increases exponentially with the network

* Corresponding author.
E-mail address: zhangjw@zzu.edu.cn

scale. For a large-flow network environment, the application deployment requirements cannot be met. This paper proposes a fast multipath jump algorithm for security constraints. The algorithm effectively combines traditional network and software-defined network characteristics to build a secure transmission path and improve network computing efficiency.

2. Research Status

The basic idea of dynamic path technology is to dynamically change the transmission paths of traffic between two sides of communication, thus enhancing the difficulty of detection and attack on specific paths for attackers.

A typical example of dynamic path technology is multi-layer overlay network protection technology based on dynamic backbone [9]. Its main idea is to build a larger external coverage network on top of the underlying network. The internal network relies on multiple internal overlay networks, providing security objectives such as encryption, dynamic routing, and configuration diversity through multi-layer overlay networks. The network's topology is logically and dynamically changed to achieve dynamic transmission paths. Literature [10] designed and implemented a dynamic defence system based on software-defined network technology, which resists network reconnaissance attacks through a combination of IP jump and path jump. However, this document does not discuss how to achieve the generation and change of fast multipath.

Later, Talipov et al. proposed a path jump method based on R-AODV (reverse AODV) [11]. With R-AODV, the source node can establish multiple paths to the destination node. It adaptively jumps to the available path during data transmission to protect the data from intrusion by malicious nodes. This method is suitable for self-organizing networks and is difficult to implement directly in wired networks.

Duan et al [12-13] proposed an active random route muting (RRM) technology. The selectable forwarding paths are calculated by satisfying the constraint that the satisfiability modulo theories (SMT) formalizes the protocol forwarding path, so that each route satisfies security, capacity, overlap, and QoS constraints. RRM has significantly increased the standard of attackers. Because of multipath traffic, an intruder must monitor all routes. Therefore, more resources are needed than the resources required to attack a single route. In addition, non-deterministic routing destroys the reconnaissance of the attack plan and wastes an attacker's resources by forcing him to blindly spread its resources across the network route. Therefore, RHM can defend detection, eavesdropping, and Dos attacks and meet end-to-end QoS characteristics. The main constraints are as follows:

Capacity constraint: Routes should not contain nodes that are already overloaded (based on node capacity) or those that do not meet the traffic bandwidth requirements.

Overlap constraint: To increase unpredictability and achieve reasonable load balancing, the overlap between routes should be less than the tolerable overlap threshold.

Safety constraint: Routes should be secured by access control policies such as firewalls; for example, if traffic must pass through a firewall, the firewall must be included in all routes.

QoS constraint: Routing should maintain the required quality, such as limited latency or hop count.

Subsequently, on the basis of RRM, literature [14] proposed an optimal jump path generation method based on a security capacity matrix and chose the optimal combination of jump path and jump cycle to maximize defence gains.

Dynamic network technology provides different secure transport servers for users with different security needs. At the same time, the attacking difficulty of the attacker on the transmission path is improved. However, dynamic network topology technology relies heavily on the robustness of the underlying network. Controlling the transfer of information between nodes, network reconfiguration, and routing increases network overhead.

3. Path Selection

The main challenge of this paper is to randomly change the path between a given source and destination address, taking into account the following limitations: (1) increase unpredictability, (2) avoid any link overload in the network (capacity limitation), and (3) satisfy QoS constraints.

A physical network can be represented as a weighted undirected graph $G = (N, L)$, where N and L respectively

represent a collection of physical switches and physical links. For each physical link $l_i \in L$, there is one available bandwidth resource $B(l_i)$, delay $D(l_i)$, jitter $J(l_i)$, and packet loss rate $S(l_i)$.

The main purpose of this paper is to build a security system that does not require optimal QoS for the generated path. Because the optimal solution will result in the bias of the selection, it is easy for an attacker to find the law of jumping. Therefore, the generation of the path is required to be more distributed and random. Therefore, the QoS requirements are also limited to the minimum QoS standards for different services. The weight selection criteria for the link in this paper is based on the available bandwidth $B(l_i)$, but the following constraints should be met for any path $R_i(1 < j < K)$ generated: $\min_{l_i \in R_j} B(L_i) \geq B$, $\sum_{l_i \in R_j} D(l_i) \leq D$, $\sum_{l_i \in R_j} J(l_i) \leq J$, $\prod_{l_i \in R_j} S(l_i) \leq S$.

According to the service requirements, B, D, J, S takes different values. This paper uses the recommended values of various service QoS parameters given by ATM, Diffserv, ITU-T, and other technical systems or standards as the basis, as shown in Table 1. The link data can be obtained by collecting the link state data collected in the Inventory node in the ODL's DataStore.

Table 1. Performance specifications of typical services

Typical business	**Packet loss *S***	**Delay *D***	**Shake *J***	**Upstream bandwidth B_{up}**	**Downstream bandwidth B_{down}**
High-speed interactive services such as online games	< 0.1%	< 200ms	< 30ms	2M	2M
Stream media	< 0.1%	< 1~2s	< 1s	2M	6Mbps/Conversation
IPTV service	< 0.1%	< 150ms	< 20ms	/	2-8M
E-commerce, confidential business	< 1%	< 1s	/	/	/
Audio and video information	< 0.1%	< 250ms	< 10ms		
Data file	< 0.01%	< 1000ms	/		
Graphics, still images	< 0.01%	< 1000ms	/		
FTP, P2P	< 0.1%	/	/		
Real-time data	< 0.01%	1~1000ms	/	512Kbps	2Mbps
Web browsing	< 1%	< 1000ms	< 1000ms	0.2Mbps	4Mbps
Telnet	< 0.1%	< 1000ms	/	/	/

In the path jump mechanism of this paper, when the controller needs to establish a path for both ends, the initial path R_1 is obtained by the Dijkstra algorithm optimized by the queue. Then, the flow table is quickly established to establish a connection. Meanwhile, a timer is set for the session. When the jump period expires, based on the initial path and the current link situation, K paths are generated using the K shortest path algorithm. Finally, one is randomly selected as the next transmission path, and a corresponding flow table is generated for delivery. In this way, the method of establishing communication in the initial path and then generating the jump path can reduce the processing delay and enhance the processing efficiency of the packet. Especially in the case of relatively large traffic, the processing of the jump path after the connection is established can significantly improve the communication establishment speed and improve the user experience. The specific path generation is shown in Algorithm 1.

Algorithm 1: Dynamic Path Jump

Input: Physical topology $G(N, L)$

Output: Flow table flow

function PATH GENERATE
  for all Packet *p* comes from the OF switch **do**
    $w[L]$ ←Calculate weights based on link information
    R_1 ← Dijkstra($G(N, L)$, $w[L]$)
    flow ←Generate a flow table according to R_1
    Release flow table *flow*
  end for
end function

```
function PATH MUTATION
 if Jump cycle to then
  R ←Based on R_1, use K shortest path algorithm to get path group
   R_j ←Randomly select a path
  flow ←Generate a flow table according to R_j
  Release flow table flow
 end if
end function
```

4. Efficient Path Generation Algorithm

The YEN algorithm calculates a single source K shortest acyclic path for a graph with non-negative edges. The algorithm was published by Jin Y. Yen in 1971. The algorithm uses the deviation path algorithm in the recursive method. The first shortest path R_1 is calculated first, and then the other K-1 shortest paths are calculated in turn. When R_{i+1} is found, all nodes on R_i except the terminating node are regarded as deviating from the node, and the shortest path from each node to the terminating node is calculated. Then, the previous path from the starting node to the deviating node on R_i is spliced to form a candidate path, and then the shortest deviation path is obtained.

This paper improves the YEN algorithm, adds capacity, overlap, and QoS constraints, and solves the shortest path that satisfies the constraint, called YEN change. The algorithm first calculates the first shortest path R_1 that satisfies the constraint and then solves the largest disjoint Z path that satisfies the constraint. Finally, the K-Z-1 path satisfying the constraint is solved based on the YEN algorithm. The specific algorithm is shown in Algorithm 2.

Algorithm 2: Dynamic Path Jump

```
Input：Physical topology G(N, L), src, dst, K
Output：K paths
function KSP
  // Determine the shortest path from source to destination.
  A[0] ← Dijkstra(Graph, src, dst);
  // Initialize the collection to store the potential k^th shortest path。
  B ← [];
  for k from 1 to K do:
    previousPath ← A[k-1];
    for each edge in previousPath.edge do:
      // Remove all links from the previous path from the diagram.
      remove edge from Graph;
    End for
    // Calculate the path from the source node to the target node.
    path ← Dijkstra(Graph, src, dst);
    if path Not empty && meets QoS constraints do:
      A[k] ← path;
    else:
      break;
  End for
  // Add the edge removed from the Graph
  restore edges to Graph;
  A[0] ↔ A[size(A)-1] exchange；
  for k from 1 to K do:
    for i from 0 to size(A[k−1])−2 do:
      spurNode ← A[size(A)-1].node(i);
      // The path from the source to the child source node i.
      rootPath ← A[size(A)-1].nodes(0, i);
      for each path p in A do:
        if rootPath == p.nodes(0,i) do:
```

```
23:             // Delete the next path of the same root path.
                remove p.edge(i, i + 1) from Graph;
24:           End if
25:         End for
26:         for each node rootPathNode in rootPath except spurNode do:
27:     // Delete the node that belongs to the root path except the child source node.
            remove rootPathNode from Graph;
28:         End for
29:         // Calculate the subpath from the child source node to the target node.
            spurPath ← Dijkstra(Graph, spurNode, dst);
30:         // The entire path consists of a root path and a subpath.
            totalPath ← rootPath + spurPath;
31:         if totalPath satisfies QoS constraints do:
32:           // Add the potential k shortest path to the heap.
              B.append(totalPath);
33:         // Add edges and nodes removed from the Graph
            restore edges to Graph;
34:         restore nodes in rootPath to Graph;
35:       End for
36:       if B is empty do:
37:         break;
38:       // Sort the potential k shortest paths by cost.
          B.sort();
39:       // Add the lowest cost path to the k shortest path.
          A[size(A)] = B[0];
40:       B.pop();
41:     End for
42:     return A;
43:   End function
```

5. Flow Consistency Problem

When the flow table is updated, the switch will have four cases: 1) does not belong to new and old paths, 2) belongs only to new paths, 3) belongs only to old paths, and 4) belongs to new and old paths. For case 1), no consideration is required. For case 2), the message is forwarded according to the new flow table. For case 3), after the new flow table is delivered, if the message arrives, it is processed according to the old flow table. If the message no longer arrives, the old flow table is deleted after Idle Timeout. For case 4), because the match is the same, the new flow table will overwrite the old flow table, so the message is processed according to the new flow table. Therefore, in the flow table update process, there are two cases of the packet transmission path: A) the packet is transferred to the new path transmission and B) the flow table packet arrives at the destination according to the old path.

For case B, the flow table has been slaved to ensure that the message is not lost during transmission. For case A, there is an inconsistency in the delivery of the flow table. If the packet arrives and the matching flow table has not been sent, the upload controller will send the packet to the next switch through the Packet out message, so that there will be no packet loss during the flow table update process. The normal transmission of the message is guaranteed.

6. Theoretical Analysis

For traffic eavesdropping in the network, assuming that the data stream transmitted during a period of time T is f , the number of link hops in time T is K. The number of nodes in the transmission path is $h_i, i \in [0, K]$, and the total number of nodes in the network is n.

This section first models the attacker before analysing the effect. A successful attack by an attacker is defined as obtaining a complete data stream f. It is assumed that the attacker has the ability to know whether the link is a target streaming link and has hopped. According to the link condition, the attacker can reselect the listening node according to the policy. The attacker can continuously scan s times in T time.

(1) Static Path

For a static link, an attacker can acquire the entire data stream f as long as it detects one of the transmission paths. Let a denote the number of links that are swept out in the s-scan, and the probability of success of the attacker is shown in Equation (1).

$$\begin{aligned} P_r(win) &= P_r(Z_s > 0) \\ &= 1 - P_r(Z_s = 0) \\ &= 1 - \binom{n-h_0}{s} / \binom{n}{s} \end{aligned} \tag{1}$$

(2) Dynamic Path

For dynamic links, the attacker must re-explore the new link of f during its jump period after each hop; otherwise, the eavesdropping fails. Therefore, the probability of an attacker's success is shown in Equation (2).

$$P_r(win) = \prod_{i=0}^{K} \left\{ 1 - \binom{n - h_i - sgn(i)}{q} \Big/ \binom{n - sgn(i)}{q} \right\} \tag{2}$$

Where $sgn(*)$ is a symbolic function and q is the maximum number of probes by the attacker during the path jump period. When $0 \prec q \prec 1$, the path jump period is greater than the detection frequency, and the attacker success probability is 0. This situation is generally not considered because of the attacker's ability to attack. In this case, the path jump rate is very fast, which requires very high performance of the controller and switch and also seriously affects the data stream transmission process. Therefore, the general default is $q \geq 1$.

Figure 1 can be obtained from Equations (1) and (2), where $n = 20$, $h_i = 40$. It can be found that as the attacker's detection ability increases, that is, the value of *k* increases, the probability that data is eavesdropped is greater. When the system path jump period is increased, the probability of an attacker's attack is gradually reduced. In the actual physical machine test process, because the path jumps, the attacker's eavesdropping traffic is incomplete and has good anti-eavesdropping effect.

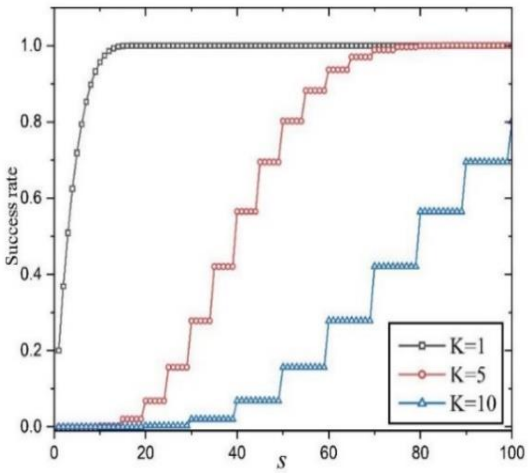


Figure 1. Path transition effect

7. Simulation Evaluation

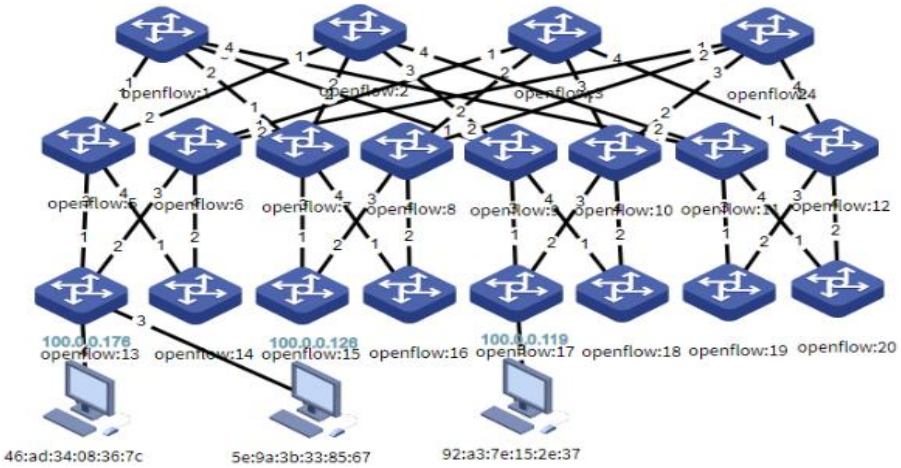
7.1. K Value Selection

The selection of the number *K* of jump paths is related to the network size. To this end, the overlap rate of the network node i is defined as δ.

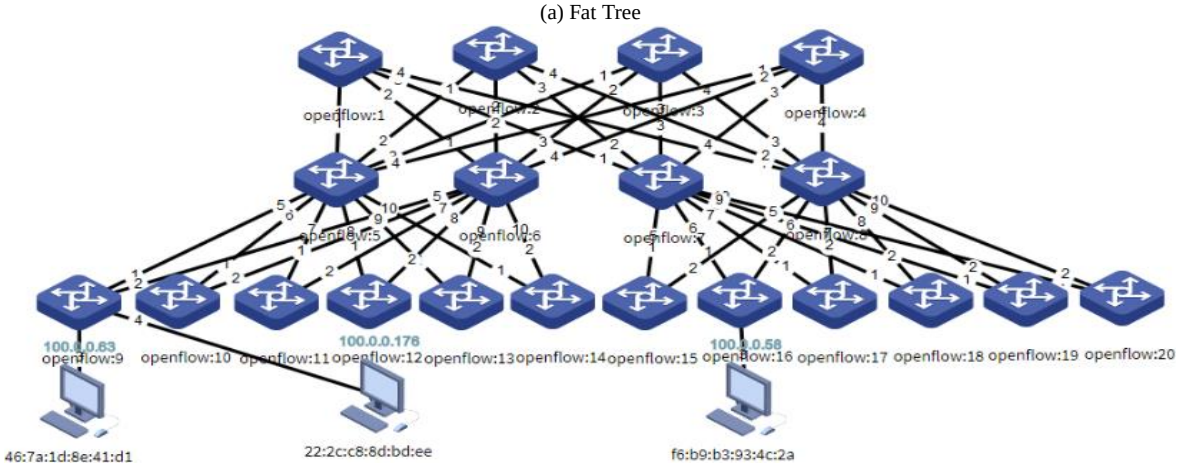
$$\delta = n_i / N \tag{3}$$

Where N is the number of times of jump in time T and n_i is the number of occurrences of node i in time T.

In theory, under an ideal network topology, the overlap rate of the network will decrease as the K value increases. However, the actual deployed network topology is classic network topology, such as Fat Tree [15] and VL2 [16]. The actual deployment network topology is shown in Figure 2.



(a) Fat Tree



(b) VL2

Figure 2. Data center network architecture

Figure 3 shows the variation of the maximum overlap rate $\delta_{\max}$ of the network with different k values under the two standard network topologies in Figure 2. It can be seen that the dynamic path jump can effectively reduce the network overlap rate. The experimental results show that the larger the value of K, the higher the path overlap rate. As the value of K increases, the number of generated path nodes begins to increase, causing some key nodes to increase in frequency. Moreover, it can be seen that the difference between K=100 and K=5 is not large. Because the calculation of the K value requires system resources, it is recommended that the K value should not be greater than half of the hop count of the generated path.

7.2. Algorithm Overhead

This section compares the SMT algorithm [12] with the proposed YEN modification algorithm. Figure 4 shows the topology in a large VL2 network. The time cost caused by the two algorithms when solving $K = 4$ shows that the time for SMT to solve constrained routing is significantly higher than that for YEN as the size of the network grows exponentially.

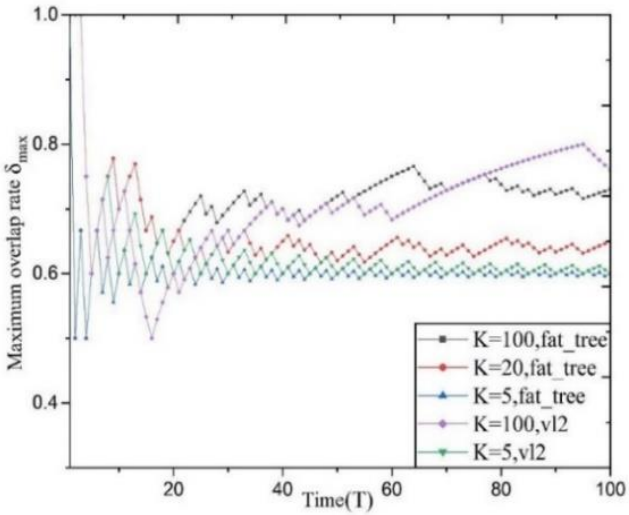


Figure 3. Network overlap rate

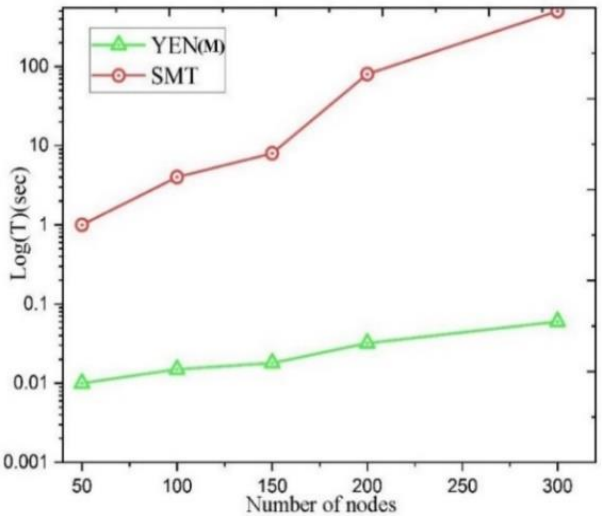


Figure 4. Path calculation time

8. Conclusions

This paper proposes a software-based active random path jump technique. Based on spatial multipath, an attacker cannot eavesdrop or implement a hijacking attack at a certain point or a link on the data stream transmission path. The existing YEN algorithm is improved. The experimental results show that the YEN modification algorithm brings great performance improvements when the optimal solution is lost.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 61672471), Henan Province University Science and Technology Innovation Team (No. 18IRTSTHN012), Plan for Scientific Innovation Talent of Henan Province (No. 184200510010), and Ph.D. Foundation of Zhengzhou Institute of Light Industry (No. 2016BSJJ041).

References

1. Q. L. Han, B. Shao, and L. J. Li, "Publishing Histograms with Outliers under Data Differential Privacy," *Security and Communication Networks*, Vol. 9, No. 14, pp. 2313-2322, 2016
2. G. R. Chen, "Problems and Challenges of Control Theory in Complex Dynamic Network Environment," *Chinese Association of Automation*, Vol. 39, No. 4, pp. 312-321, 2013
3. S. Tarciana, G. Reinaldo, O. Luciana, C. Igor, S. Djamel, and J. Martin, "OSPF-AN: An Intra Domain Routing Protocol for Ambient Networks," in *Proceedings of International Conference on Advanced Communication Technology*, Vol. 1, pp. 212-217,

2008
4. Y. F. Lei, "Application Research of OSPF Dynamic Routing Protocol in Campus Network," *Computer CD-ROM Software and Application*, Vol. 16, pp. 299-300, 2014
5. Z. J. Feng and Q. Zhu, "Enterprise Network Design based on OSPF (Open Shortest Path First) Routing Protocol," *Computer Knowledge and Technology*, Vol. 14, No. 18, pp. 16-18, 2018
6. Q. Hua, Y. S. Li, and J. Liu, "A CCN Centralized Control Cache Decision Method based on SDN," *Telecommunications Science*, Vol. 33, No. 5, pp. 12-20, 2017
7. Z. Li, "Research on Control Plane Availability and Reliability based on SDN Architecture," Beijing University of Technology, 2015
8. D. B. Qi, X. G. Wang, and X. Fu, "Exploration and Research on Technical Difficulties of Multi-Path System in Cross-Organizational Trials," *Drug Evaluation Study*, Vol. 40, No. 10, pp. 1372-1377, 2017
9. J. D. Touch, G. G. Finn, Y. S. Wang, and L. Eggert, "DynaBone: Dynamic Defense using Multi-Layer Internet Overlays," in *Proceedings of DARPA Information Survivability Conference and Exposition*, pp. 22-24, IEEE, US, 2003
10. Y. Chen, H. C. Hu, and G. Z. Cheng, "Design and Implementation of Software-Defined Intranet Dynamic Defense System," *Acta Electronica Sinica*, Vol. 46, No. 11, pp. 2604, 2018
11. E. Talipov, D. Jin, and J. Jung, "Path Hopping based on Reverse AODV for Security," in *Proceedings of Asia-Pacific International Conference on Network Operations and Management*, pp. 574-577, Springer-Verlag, 2006
12. Q. Duan, E. Al-Shaer, and H. Jafarian, "Efficient Random Route Mutation Considering Flow and Network Constraints," in *Proceedings of 2013 IEEE Conference on Communications and Network Security*, pp. 260-268, IEEE, 2013
13. W. Lan, J. X. Wang, M. Li, W. Peng, and F. X. Wu, "Computational Approaches for Prioritizing Candidate Disease Genes based on PPI Networks," *Tsinghua Science and Technology*, Vol. 20, No. 5, pp. 500-512, 2015
14. G. L. Sun, T. Chen, Y. Y. Su, and C. L. Li, "Internet Traffic Classification based on Incremental Support Vector Machines," *Mobile Networks and Applications*, Vol. 23, No. 4, pp. 789-796, 2018
15. M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, pp. 63-74, 2008
16. A. Greenberg, J. R. Hamilton, and N. Jain, "VL2: A Scalable and Flexible Data Center Network," *Association for Computing Machinery*, Vol. 39, No. 4, pp. 51-62, 2009

Wanwei Huang received his Ph.D. in communication and information systems from PLA Information Engineering University in 2009. He is currently a lecturer at Zhengzhou University of Light Industry. His research interest is communication engineering.

Yang Chen is a master's student at the National Digital Switching System Engineering Technology Research Center. His research interests include cloud computing, SDN, and network security.

Jianwei Zhang received his Ph.D. in computer application technology from PLA Information Engineering University in 2010. He is currently a professor at Zhengzhou University of Light Industry. His research interests include broadband information networks and network security.

Chunfeng Du is a master's student in the School of Computer and Communication Engineering at Zhengzhou University of Light Industry. His research interests include network engineering and big data.

Sunan Wang is an associate professor in the Electronic and Communication Engineering College at Shenzhen Vocational and Technical College. His research fields are network system architecture, data flow analysis, and information security.