Technical Design of an Ad-Hoc Network Topology Simulation System

Zhongyu Yin[1(✉)] and Shuo Shi[1,2]

[1] School of Electronic and Information Engineering, Harbin Institute of Technology, Harbin 150001, Heilongjiang, China
a87557766@163.com, crcss@hit.edu.cn

[2] Peng Cheng Laboratory, Network Communication Research Centre, Shenzhen 518052, Guangdong, China

Abstract. In order to study the adaptation of Ad-hoc networks to different topologies at the network layer, this paper designs a platform based on Raspberry PI hardware to meet the demand of hardware-in-the-loop networking simulation of Ad-hoc networks on the desktop. Firstly, this paper will demonstrate the importance of physical topology scenario simulation in Ad-hoc network technology development and deployment. Secondly, this paper will propose a physical topology scene control and performance monitoring scheme based on the Netfilter module used for network layer data processing in Linux system. This scheme will be programmed and deployed to Raspberry PI node, and then automatically executed by a set of graphical control system on Ubuntu 20.04 host. Finally, this paper will rely on the above system to simulate the representative network topology in certain practical applications, and test the performance indexes of OLSR and its evolved version OLSR V2 respectively, so as to evaluate the performance of the two protocols in specific systems.

Keywords: Ad-hoc · Network Simulation · OLSR

1 Source and Purpose of the Project

The proliferation of electronic devices and corresponding communication technologies has become a sign of The Times we live in. Taking the ground mobile communication network as an example, people can easily achieve long-distance, low-delay and high-rate wireless communication through mobile phones and base stations all over the continent. In general, such networks, which rely on base stations for access and grids for power supply, are convenient, fast and reliable. However, under special circumstances, the mobile communication network that depends on the infrastructure is deficient in reliability [1]. Relating to this topic, on the other hand, self-organizing network technology as a kind of network mode is different from the ground mobile communication network, does not depend on infrastructure [2], can adapt to different topologies and jump forward network technology, more because of its anti-damage ability showed its emergency communication [3], military police and other special value in the field of communications [4].

A. Li et al. (Eds.): 6GN 2022, LNICST 504, pp. 26–37, 2023.
https://doi.org/10.1007/978-3-031-36011-4_3

Therefore, the embodiment of this technology at the network layer, that is, the adaptation of Ad-hoc network routing protocol to different topologies, is worth further research and testing.

The so-called Wireless Ad-hoc Network refers to a Wireless communication Network with the following characteristics [5]:

(1) No center. All nodes participating in the network have equal rights in principle, and there is no dispatching node as the control center, which is also the source of strong damage resistance of Ad-hoc networks.
(2) Independent networking. After the nodes are powered on and running, they independently form an independent network according to the pre-written protocol logic.
(3) Multi-hop routing. The communication between two nodes outside the direct communication distance can be relayed and forwarded by multiple ordinary intermediate nodes. Each intermediate node is called a hop, and the whole route process can be multiple hops.
(4) Dynamic topology. Firstly, topology generally refers to network topology in the field of communication, which belongs to the category of graph theory in theory. It is a concept that abstracts the physical devices participating in the network as points and the direct links between devices as chains to describe the connection relationship between physical devices in the network. However, in Ad-hoc networks, nodes can join or exit at any time, and the original direct connection between any two neighbor nodes may be interrupted due to factors such as movement or occlusion, which leads to dynamic network topology [6].

The term topology needs a little more explanation here. In this paper, the term topology can be divided into two categories:

(1) Physical topology, especially the communication links and topological associations in the physical channels of actual communication;
(2) Protocol topology, especially the network topology detected by routing protocols.

In normal cases, the protocol topology detected by the routing protocol of Ad-hoc networks is the actual topology, which is also the basis for routing protocol to select routes. In the simulation environment, the physical topology can be easily specified by simulation scenarios, but in the experiment of the actual system development stage, in order to test the performance of routing protocols under different topologies [7], it is necessary to physically create the specified topology by means of pulling distance, occlusion, shielding, etc., which is extremely inconvenient.

On this basis, the research focus of this paper is to design a platform based on Raspberry PI hardware, and realize the simulation of Ad-hoc networking in different topology scenarios on the desktop.

Therefore, in the design of this paper, a new topology is artificially added between the above two topologies, that is, the simulation scene topology. The system topology association after the addition is shown in Fig. 1:

As shown in Fig. 1, in a development environment, the physical topology of each simulated node can be considered fully connected due to its close proximity. Based on this fully connected physical topology, different simulated scene topologies can be artificially

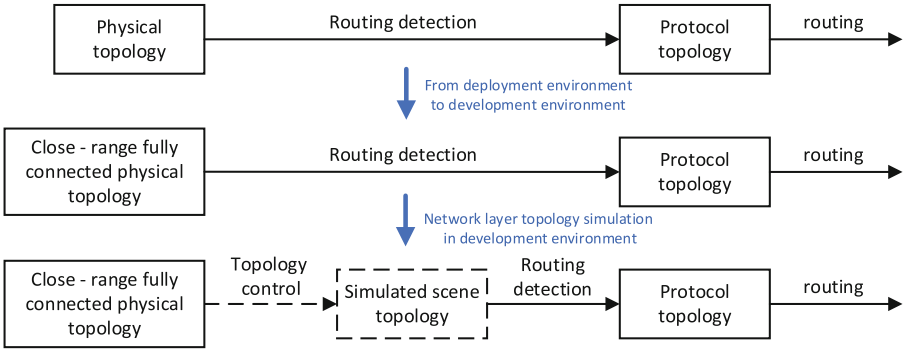

Fig. 1. The connection of several topologies in this article.

created by filtering incoming packets in the Linux network layer. For routing protocols, the simulated scene topology generated by masking is equivalent to the physical topology, so it can replace the pull test to simulate the behavior of routing protocols in different topology scenarios and test their performance, which is also the meaning and goal of this design topic.

2 Overview of Topology Simulation Principles

2.1 Description of Topology

A common mathematical description of network topology is the adjacency matrix. Adjacency matrix is a matrix representation of graph in graph theory.

In this paper, since there is no unidirectional link in the network topology between nodes, the network topology can be abstracted as an undirected graph $G(V,E)$, and its matrix expression, that is, the undirected adjacency matrix G [8], is a square matrix with the following properties:

(1) G is a real symmetric square matrix of order N with diagonal element 0;
(2) Each element of G can be 0 or 1 only;
(3) G_{ij} represents the topological connection between node i and node j. If the topology exists, that is, there is a direct link, then $G_{ij} = 1$;
(4) If no direct link exists, $G_{ij} = 0$. In this case, the corresponding data packets need to be masked in the system.

For example, in the case of three points, $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$ said the connection topology relationship, and $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ said chain topology relationship.

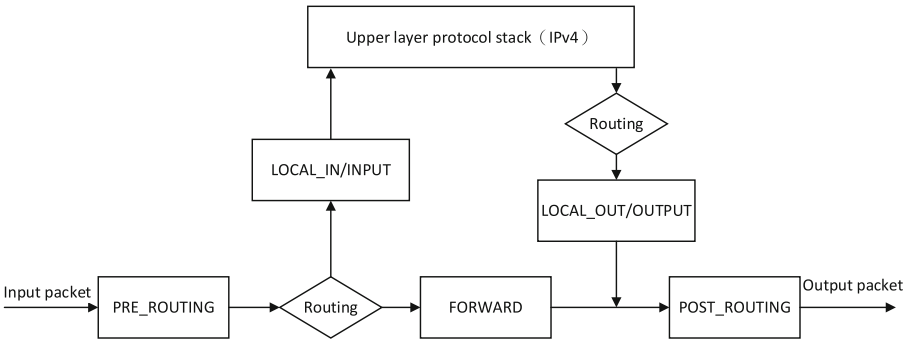

Fig. 2. Location of Netfilter/iptables processes and Hooks/linked lists.

2.2 Topology Control and Performance Monitoring Principle Based on Netfilter

Aiming at the network layer topology control method, this paper proposes a topology control scheme based on Netfilter module of network layer of Linux system:

In principle, Netfilter/iptables is a processing mechanism with firewall function introduced into network layer data flow processing after Linux2.4.x [12]. From the perspective of implementation, Netfilter/iptables is a subsystem of Linux kernel. It consists of three parts: Netfilter framework, iptables linked list and iptables command line tool. Netfilter is fully compatible with the IP protocol and provides users with functions such as filtering, address translation, and processing before data is delivered to the IP protocol for processing [9]. Netfilter sets a total of 5 hook points on the IPv4 protocol stack, and several functions can be set on each hook point according to the priority to realize the processing of packets flowing through. Their positions are shown in the Fig. 2.

On this basis, the topology control tool selected in this paper is the iptables command line tool provided by Netfilter in the application layer. Simply put, the iptables command line tool and the principle behind it abstracts each Hook node through which Netfilter data flows as a chain, and the data processing functions in the chain are encapsulated in the form of a rule table that can be added and modified. In particular, iptables defines up to four tables with different functions on each chain, namely:

(1) Filter table: responsible for filtering the packets flowing through according to the rules to realize the firewall function;
(2) NAT table: Network address translation to realize network address translation;
(3) Mangle table: disassemble packets, modify them and repackage them;
(4) RAW: Disables the connection tracing mechanism on the NAT table.

It can be seen that the implementation of the filter function is the filter table. However, since the PRE_ROUTING chain does not provide filtering function for data security, discarding data can only be transferred to the INPUT chain and FORWARD chain. The reason why the data in the OUTPUT chain is not filtered is that the specific destination IP or MAC address cannot be identified due to the presence of UDP broadcast packets.

After the above analysis, it can be determined that the data on the INPUT and FORWARD chains is all the data flowing into the local machine. However, for the sake of rigor, an additional question needs to be answered, that is, the data flowing through

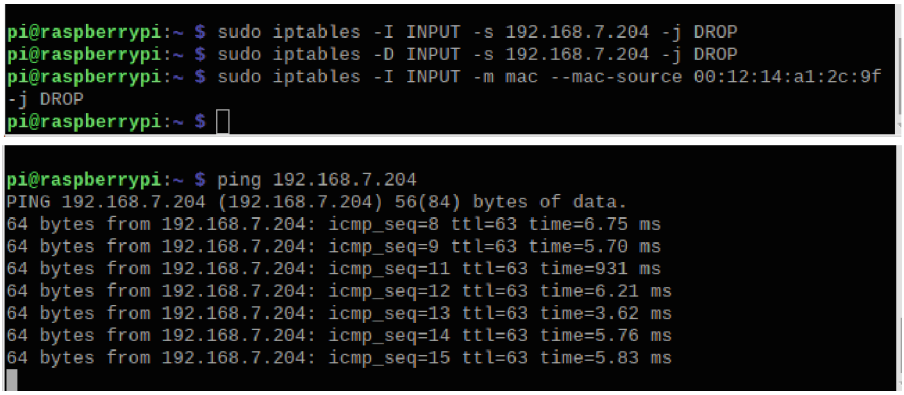

Fig. 3. Topology control based on MAC addresses.

the INPUT chain is routed and the destination address is the data on the local machine. Does this mean that the INPUT chain is already behind the routing protocol process?

However, it can be found that protocol software such as OLSRD and OLSRD2 run completely on the application layer by combining with the related research on the implementation principle of typical OLSR routing protocol. In the process of running, it will act as a background application of the system to maintain the routing table of the system through UDP protocol, such as broadcast detection, instead of directly modifying the data flow.

The experiment also proves the validity of this topology control principle for OLSR protocol. In Iptables tool, developers can mask nodes by physical address, that is, MAC address. The command format is: "sudo iptables -A INPUT -m mac --mac-source <MAC Address>-j DROP" [10].

The topology control test results are shown in Fig. 3:

In addition to filtering, the linked lists of Iptables also count the number of packets in bytes and the size of packets in bytes when the rules match successfully. Iptables starts counting from the time a rule is added and stops until the rule is deleted or the result is emptied when an instruction with a "-z" argument is received. When receiving a command with the "-nvx-l" parameter, iptables will output the statistics to the console terminal through the standard input-output pipeline. With this method, we can monitor the link performance indicators in real time by adding matching rules for MAC addresses and port numbers, and clean up the data after the simulation.

3 Design of Ad-Hoc Network Topology Simulation System Based on Raspberry PI

3.1 System Architecture and Development Environment

After completing the research on topology representation and control principle in the previous chapter, the next topic will shift from theoretical analysis to the design and engineering implementation of the overall topology simulation system. The work of this

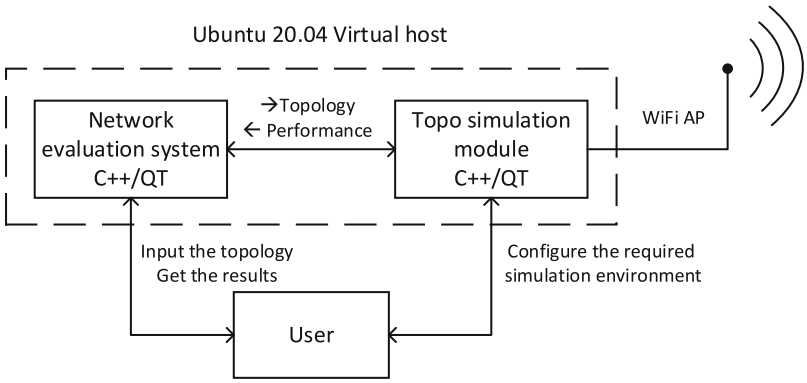

Fig. 4. Topology simulation system scheme diagram of the host.

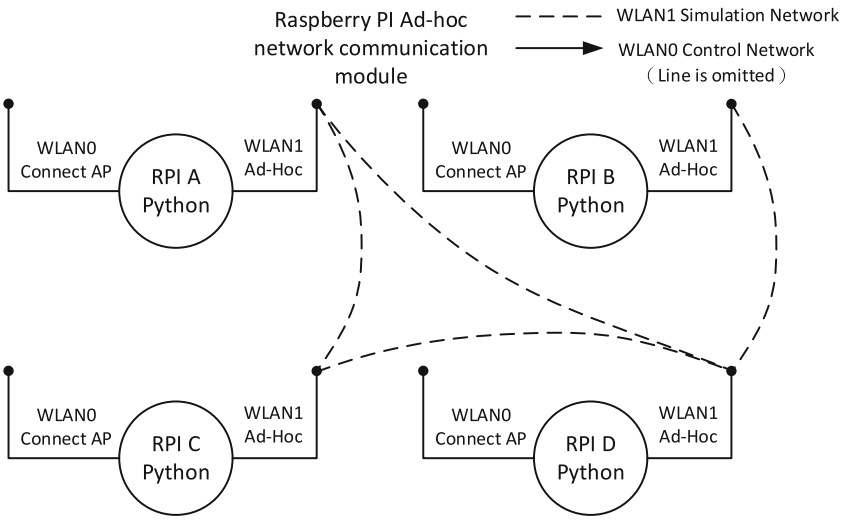

Fig. 5. Topology simulation system scheme diagram of the Raspberry PI communication module.

chapter is how to integrate the preceding independent functions into a system through a set of programs running on the host and Raspberry PI, so that users only need to interact with the graphical interface of the host, and can realize the topology control, communication behavior and real-time performance data return on the node with a simple click. To this end, I first give the specific scheme diagram:

Figure 4 and Fig. 5 show the scheme diagram of the control host program and the communication simulation module of Raspberry PI Ad-hoc network respectively. Two features of this design can be seen from it:

First, this design is not completely merged with the upper network performance evaluation system, but exists in the form of loadable modules. The upper-layer network evaluation platform is responsible for providing the interface for users to deploy topology

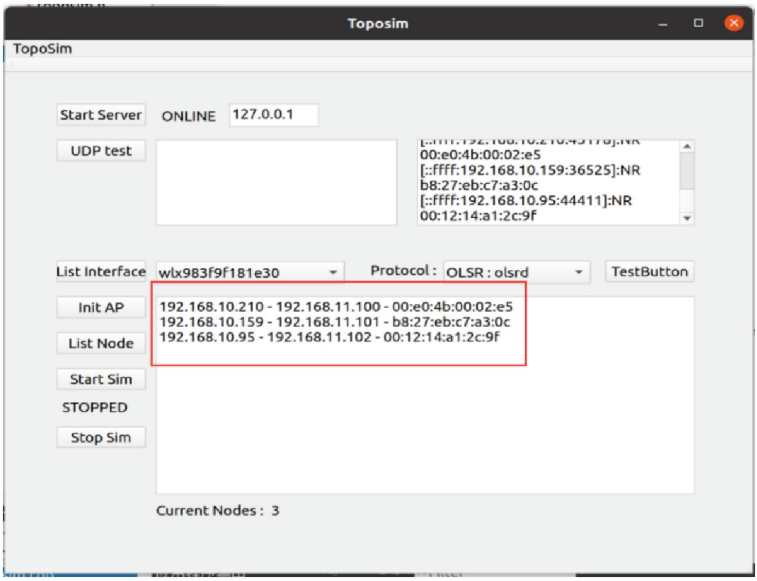

Fig. 6. Nodes probe and simulate network IP assignment results.

and dynamically detect performance statistics, while the topology simulation module acts as a computing engine and only provides necessary control functions related to physical simulation.

Second, as shown in Fig. 5, in order to simultaneously realize the transmission of control information and performance data between the host and Raspberry PI, as well as the Ad-hoc network simulation communication between Raspberry PI and Raspberry PI, two sets of networks are constructed on Raspberry PI with two network cards in this system:

(1) Raspberry PI built-in network card WLAN0: used to connect to the open WiFi hotspot of the host, forming a "control network". On this basis, the two ends of the program to transfer control data and performance data to each other by UDP protocol;
(2) Raspberry PI external network adapter WLAN1: The model of network adapter chip selected in this design is RT3070, which can run in AP mode and Ad-hoc mode after testing. This NIC is used to run Ad-hoc networking protocols to form an analog network among raspberry Pies.

The purpose of adopting the two-layer network scheme in this design is mainly to consider the IP assignment problem when the Ad-hoc network protocol is running. Before the traditional Ad-hoc network devices go online, they need to be configured with different IP addresses for each node in the same network segment. In real objects, this means that different nodes are equipped with different firmware programs, which requires manual configuration one by one. Expanding the number of nodes is time-consuming and laborious, but not flexible and convenient. After adopting the two-layer

network, the DHCP service deployed by the host when establishing the WiFi hotspot enables each Raspberry PI with identical firmware to get automatically assigned IP address on the control network, and then respond to the node probe message of the host in this network to get the temporary simulated network IP address assigned by the host.

The control interface of the host is as shown in Fig. 6, this program interface is divided into three display boxes, in which the upper left is used to display key process prompts and error information, the upper right is the UDP communication data status monitoring bar, and the lower display bar is used to display the current scanned physical node status. In this figure, three nodes have been simulated, and their information from left to right are respectively the IP address of the control network, the assigned IP address of the simulated network and the MAC address of the network adapter used for simulation.

Up to now, the topology simulation system has been built, and the simulation test of OLSR and OLSR V2 protocols can be carried out based on this system.

4 Test the Performance of OLSR and OLSR V2 Routing Protocols

4.1 Scenario Classification and Deployment

The first step in the simulation is to specify a specific scenario. In this article, the test scenarios are divided into three categories:

(1) Equal networking:

The equal networking scenario is a scenario with fully connected physical topology between nodes. All nodes are neighbors and no additional topology control is required. The adjacency matrix of the equal networking is: $\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$, which can test the single-hop performance.

(2) Hierarchical networking:

In this scenario, networking devices are divided into upper and lower layers, and the two layers are independently networked in Ad-hoc networking mode. Generally, the lower layer network uses a node in the upper layer network as the default forwarding node, making the lower layer network not completely equal. The adjacency matrix of the hierarchical network is as follows: $\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$, which can test the two-hop performance.

(3) Multi-hop relay forwarding:

The multi-hop relay is generally used to communicate through the hop-by-hop relay of multiple Ad-hoc network nodes when the physical distance between the sender and the receiver is too long or the direct link is blocked or interfered. The adjacency matrix

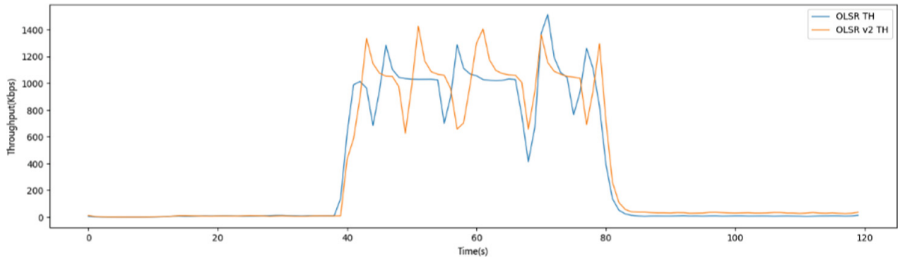

Fig. 7. Throughput curve of the egalitarian networking scenario.

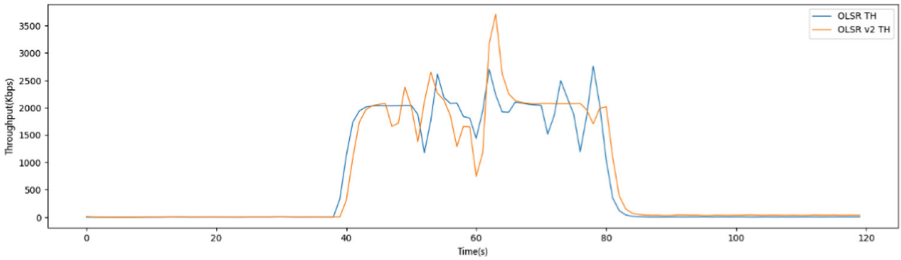

Fig. 8. Throughput curve of hierarchical networking (minimum unit) scenario.

of the multi-hop relayed forwarding scenario is as follows: $\begin{bmatrix} 0\,1\,0\,0 \\ 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 0\,0\,1\,0 \end{bmatrix}$, which can test the triple hop performance.

4.2 The Test Results

Throughput. In this section, the simulation results of throughput performance index of OLSR protocol and OLSR V2 protocol are analyzed qualitatively according to their trends in different scenarios. The measured data in each scenario are shown in Fig. 7, 8 and 9.

Considering the impact of link congestion and performance fluctuation, the throughput curves in this chapter are averaged within three seconds. As shown in the figure above, when the hardware performance meets the test service rate, the throughput statistics obtained by OLSR and OLSR V2 at the network layer in the same scenario are approximately the same. During the execution of the test service, the typical throughput in the three scenarios increases in turn, and the ratio is approximately 1:2:3, which is consistent with the global throughput generated by the 1Mbps test service in the three scenarios of 1-hop, 2-hop and 3-hop, which can prove the effectiveness of the design of the topology simulation system from the side.

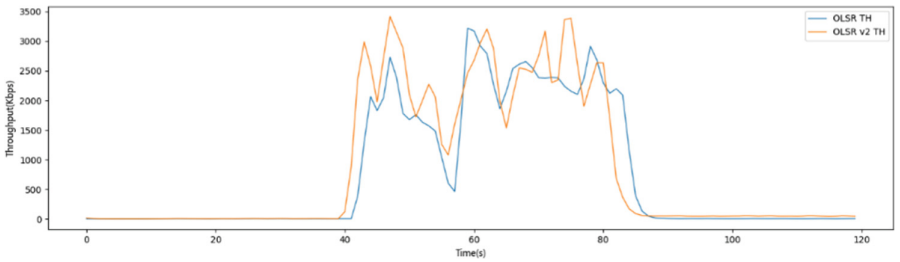

Fig. 9. Throughput curve of the multi-hop relay forwarding (three-hop) scenario.

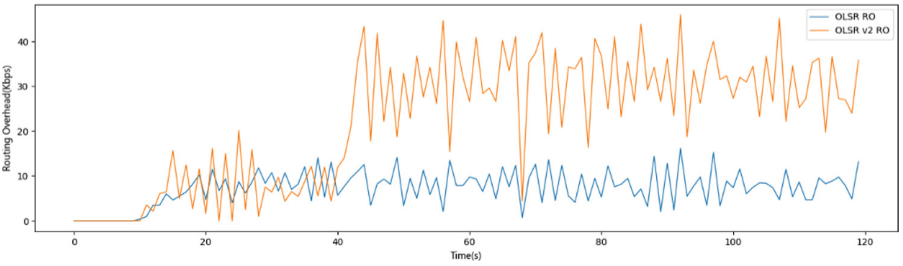

Fig. 10. Routing cost curve in egalitarian networking scenario.

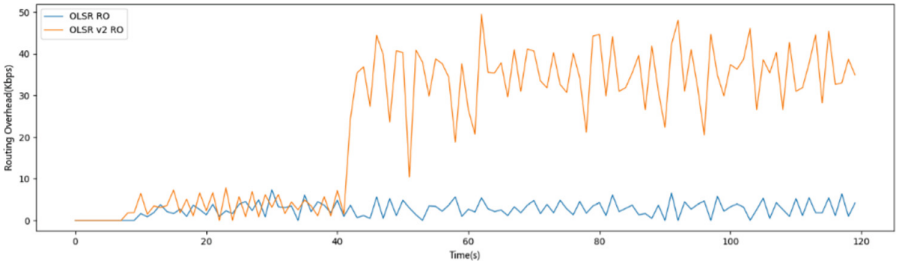

Fig. 11. Routing cost curve in hierarchical networking (minimum unit) scenario.

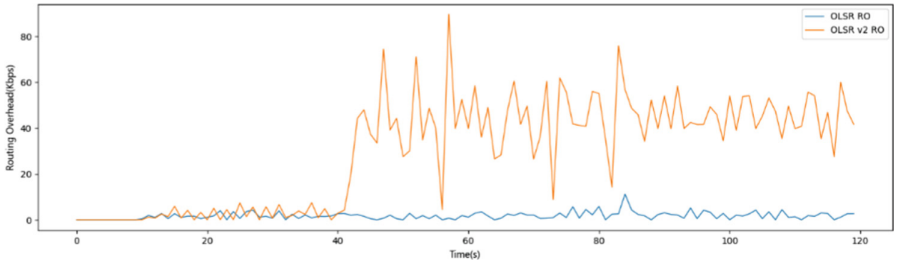

Fig. 12. Route cost curve in the multi-hop trunk forwarding (3-hop) scenario.

Rrouting Overhead. In this section, the simulation results of routing cost index will be qualitatively analyzed in different scenarios. The measured data are shown in Fig. 10, 11 and 12.

As can be seen from the figure, the routing packet sending of OLSR V2 seems to be significantly close to the traffic load. That is to say, only when the simulation time reaches 40s and a large number of traffic demands are generated, OLSR V2 will send a large number of packets to maintain the routing.

5 Conclusion

In this paper, the author theoretically discusses the implementation of topology simulation scheme, and according to the Linux operating system kernel, the Netfilter module has made research and analysis, put forward a set of topology control scheme based on Netfilter application layer interface iptables tool. After analysis and experiment, the author makes clear the meaning of iptables linked list rules, the way of node identification in the process of topology masking, and the universality of this scheme for the implementation of the selected protocol in this paper. At the end of this paper, using the developed system, the author conducted a hardest-in-kind simulation of OLSR and OLSR V2 protocols on the basis of control variables for three typical scenarios in Ad-hoc networks, namely, equal networking, hierarchical networking (minimum network unit) and multi-hop relay forwarding, and verified the effectiveness of the system designed in this paper.

Acknowledgement. This work is supported by the National Natural Science Foundation of China under Grant 62171158 and Research Fund Program of Guangdong Key Laboratory of Aerospace Communication and Networking Technology under Grant 2018B030322004.

References

1. Lei, L.: Application of wireless Ad-hoc network technology in emergency communication network. Heilongjiang Sci. **10**(14) (2019)
2. Ajith Kumar, S.P., Sachdeva, R.: Wireless adhoc networks: performance analysis considerations for AODV routing protocols. In: 2019 6th International Conference on Computing for Sustainable Global Development, INDIACom, pp. 140–144. IEEE (2019)
3. Leite, J.R.E., Martins, P.S., Ursini, E.L.: Planning of AdHoc and IoT Networks under emergency mode of operation. In: 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON, pp. 1071–1080. IEEE (2019)
4. Miao, Y., Sun, Z., Wang, N., Cruickshank, H.: Comparison studies of MANET-satellite and MANET-cellular networks integrations. In: 2015 International Conference on Wireless Communications & Signal Processing, WCSP, pp. 1–5. IEEE (2015)
5. Hanzo, L., Tafazolli, R.: A survey of QoS routing solutions for mobile ad hoc networks. IEEE Commun. Surv. Tutor. **9**(2), 50–70 (2007)
6. Toutouh, J., Garcia-Nieto, J., Alba, E.: Intelligent OLSR routing protocol optimization for VANETs. IEEE Trans. Veh. Technol. **61**(4), 1884–1894 (2012)

7. Taleb, T., Sakhaee, E., Jamalipour, A., Hashimoto, K., Kato, N., Nemoto, Y.: A stable routing protocol to support ITS services in VANET networks. IEEE Trans. Veh. Technol. **56**(6), 3337–3347 (2016)
8. Min, Z., Jiliu, Z.: A new dynamic routing protocol of wireless ad hoc network. In: 2009 Asia-Pacific Conference on Computational Intelligence and Industrial Applications, PACIIA, pp. 447–450. IEEE (2009)
9. Wang, B., Lu, K., Chang, P.: Design and implementation of Linux firewall based on the frame of Netfilter/IPtable. In: 2016 11th International Conference on Computer Science & Education, ICCSE, pp. 949–953. IEEE (2016)
10. Voronkov, A., Martucci, L.A., Lindskog, S.: Measuring the usability of firewall rule sets. IEEE Access **8**, 27106–27121 (2020)