LETTER

DVNR: A Distributed Method for Virtual Network Recovery

Guangyuan LIU[†a)], ***Member and*** **Daokun CHEN**[††], ***Nonmember***

SUMMARY How to restore virtual network against substrate network failure (e.g. link cut) is one of the key challenges of network virtualization. The traditional virtual network recovery (VNR) methods are mostly based on the idea of centralized control. However, if multiple virtual networks fail at the same time, their recovery processes are usually queued according to a specific priority, which may increase the average waiting time of users. In this letter, we study distributed virtual network recovery (DVNR) method to improve the virtual network recovery efficiency. We establish exclusive virtual machine (VM) for each virtual network and process recovery requests of multiple virtual networks in parallel. Simulation results show that the proposed DVNR method can obtain recovery success rate closely to centralized VNR method while yield ~70% less average recovery time.

key words: *network virtualization, distributed virtual network recovery, parallel, recovery efficiency*

1. Introduction

In today's social life, the Internet is playing an increasingly important function. The number of users on the Internet has grown exponentially each year. However, as the most widely distributed public network, the huge scale and coverage of the Internet poses obstacles to the introduction of new network technologies. In order to overcome the ossification of the Internet, the network virtualization [1], [2] is proposed as an innovative technology. This technology provides an effective way to build virtual network that allows multiple network architectures and applications to run simultaneously on the same substrate network. Virtual networks can support different kinds of network topologies, network services, and network experiments. However, each network application provided by the virtual network is actually provided by the substrate network devices [3]. These virtual networks and substrate devices must communicate in a reliable manner to ensure end-user requirements and QoS.

Once the virtual network fails, its corresponding network service will also be interrupted, resulting in the end-user needs not being met. Therefore, how to recovery the virtual network effectively is a key challenge to the substrate network providers. The previous VNR methods are mostly based on the virtual network remapping [4]. When the substrate network fails, the substrate network management device remaps the failed virtual network to the available substrate devices. The traditional virtual network construction algorithm (e.g. [5]–[9]) provides a theoretical basis for VNR. Furthermore, these methods assume that there is a centralized entity that has a global view of the entire network, which contains information about all network nodes and network links. Based on this assumption, this centralized entity can obtain real-time network configuration information and helps the recovery of the virtual network. However, the real network environment is complex, dynamic, and changeable. The number of substrate devices in the network is huge and the structure is diverse. Therefore, centralized processing methods will face scalability limitations, barriers to information updates, and the impact of high latency. For example, if multiple virtual networks fail, their recovery processes are usually queued according to a specific priority, which may increase the average waiting time of users. Especially for real-time services, these services can greatly affect functionality due to long periods of unresponsiveness. Therefore, distributed VNR method may be more adapted to the needs of Internet development.

In this letter, we propose a novel VNR method called DVNR. The DVNR is based on the distributed idea and process recovery requests of multiple virtual networks in parallel. We establish exclusive virtual machine (VM) for each virtual network. The virtual network's recovery request is processed by its own virtual machine. When the network resource owned by the local virtual machine is insufficient to recover the virtual network, the virtual machine can forward the recovery request to the adjacent virtual machine and perform resource consolidation to complete the entire recovery process. The simulation results show that the DVNR can obtain recovery success rate closely to centralized VNR method while yield ~70% less average recovery time. We believe it will make sense for the network service with real-time requirement and will make a complementary to the previous studies on the recovery method of virtual network.

2. Method Design

In this section, we give the details of distributed adaptive virtual network recovery method. In general, the design goals of method are as follows: (1) Restore the failed virtual networks in parallel; (2) Divide the substrate network into several subnets in units of virtual machines, periodically detecting substrate network failures and performing VNR

Manuscript received April 1, 2020.
Manuscript revised July 6, 2020.
Manuscript publicized August 26, 2020.

[†]The author is with School of Information Science and Technology, Shijiazhuang Tiedao University, Shijiazhuang, Hebei 050043, China.

[††]The author is with Institute of Software Chinese Academy of Sciences, Beijing 100190, China.

a) E-mail: gyuanliu@163.com

DOI: 10.1587/transinf.2020EDL8050

Copyright © 2020 The Institute of Electronics, Information and Communication Engineers

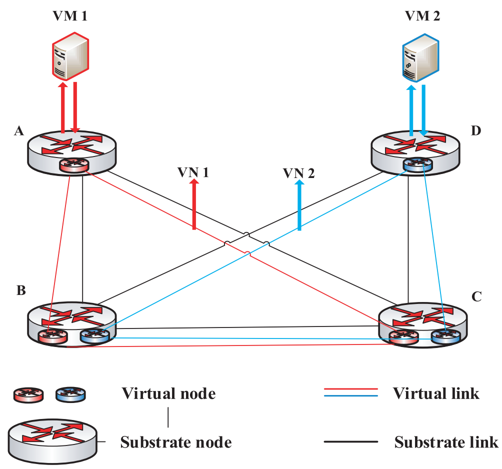

Fig. 1 DVNR overview

autonomously; (3) Each virtual machine of virtual network can exchange information and merge resources with the adjacent virtual machine to help the virtual network recovery.

2.1 DVNR Overview

Figure 1 gives an overview that the substrate network uses the DVNR method to recover virtual networks. In the virtualization environment, the heterogeneous virtual networks can share the substrate network resources, such as node capacity and link bandwidth. There are two virtual networks (e.g. VN_1 and VN_2), which map onto the substrate network in Fig. 1. We define the virtual networks that share the substrate network resources (e.g. substrate node or substrate link) as adjacent virtual network. Obviously, VN_1 and VN_2 share substrate node B, substrate node C, and substrate link BC. Therefore, they are adjacent. At the same time, we select two substrate nodes associated with VN_1 and VN_2 to run the virtual machine of two virtual networks (e.g. A, D in the figure). When VN_1 and VN_2 fail because of the substrate network failure, DVNR can complete the recovery of the virtual network according to predefined operational rules in virtual machine.

2.2 Virtual Machine Information

The virtual machine (e.g. VM_1 and VM_2) stores global information with corresponding virtual network (e.g. VN_1 and VN_2), including:
(1) Virtual network topology $G^v = (N^v, L^v, C^v_N, C^v_L)$, where N^v represents the set of virtual node, L^v represents the set of virtual link, C^v_N represents the capability requirement of virtual node, and C^v_L represents the bandwidth requirement of virtual link.
(2) The substrate network sub-graph $G^p_{sub} = (N^p_{sub}, L^p_{sub}, C^p_N, C^p_L)$, which N^p_{sub} represents a set of substrate nodes, L^p_{sub} represents a set of substrate links, C^p_N represents the processing power of the substrate node, and C^p_L represents the bandwidth of the substrate link. For example, the sub-graph information stored in the virtual machine VM_1 about VN_1 is $G^p_1 = (\{A, B, C\}, \{\overline{AB}, \overline{BC}, \overline{AC}\}, C^p_N, C^p_L)$ in Fig. 1.
(3) The virtual machine location index set of adjacent virtual network $I = \{I_{VM_1}, I_{VM_2}, \ldots, I_{VM_n}\}$ where I_{VM_i} $(1 \leq i \leq n)$ represents the physical location index of the virtual machine.

The information of all the virtual networks is stored on each virtual machine in a distributed manner. Compared with centralized information storage, distributed information storage averages the storage space and improves the access efficiency of information. Each virtual machine has a part of independent information and a part of shared information, which facilitates information interaction between virtual machines.

2.3 Virtual Network Recovery Request Forwarding Mechanism

In general, due to the limitation of the local substrate network resources, the DVNR needs to forward the recovery request of the virtual network between the virtual machines, and complete the recovery of the virtual network by merging the substrate network resources. Therefore, a sophisticated message forwarding mechanism is essential. The message forwarding mechanism of DVNR includes three parts: forwarding strategy, subnet merging and message feedback:
(1) Forwarding strategy: DVNR use virtual network recovery request forwarding strategy with breadth-first. The breadth-first policy can forward the virtual network recovery request to all the adjacent virtual machines until the hop limit is reached, or each possible forwarding path can no longer be extended.
(2) Subnet merge: DVNR uses the strategy of merging subnet resources to improve the success rate of virtual network recovery in the process of recovery request forwarding. For example, assume a recovery request req_0 from a virtual machine of a virtual network VN_0 has been forwarded k times (that is, req_0 is forwarded by the virtual machine of the virtual network $VN_1, VN_2, \ldots, VN_{k-1}$ in turn). When req_0 is received by the virtual machine of the next virtual network VN_k, the virtual machine of VN_k will be able to obtain the substrate subnet resources owned by the previous k virtual network from VN_0 to VN_{k-1} when attempting to recover the virtual network VN_0.
(3) Message feedback: If the virtual network successfully recovers by forwarding the recovery request and merging the subnet resources, the virtual machine that completes the virtual network recovery will feed back a successful recovery message to its previous hop virtual machine. After the message is fed back to the previous hop virtual machine, the virtual machine continues to feed back the information to its previous hop virtual machine. This process continues until the feedback message arrives at the source virtual machine that sent the virtual network recovery request. If the virtual network fails to be restored when the recovery request of the virtual network reaches the forwarding hop limit, the current virtual machine will feed back a failure recovery message hop by hop in the same way on the message forwarding path

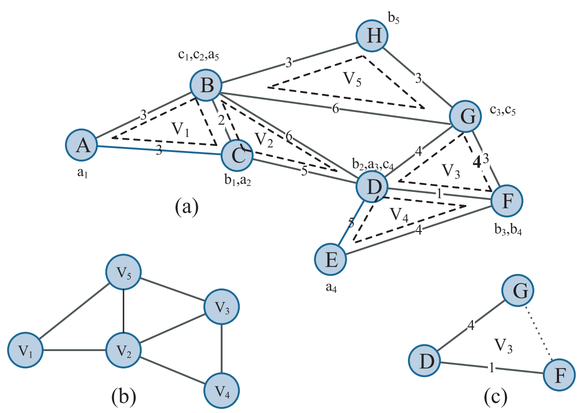

Fig. 2 The substrate network and virtual machine logical topology

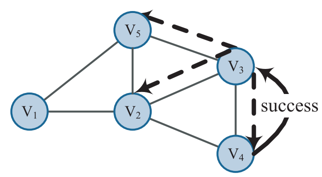

Fig. 3 Breadth-first message forwarding sequence

until the message arrives at the source virtual machine.

2.4 DVNR Virtual Network Recovery Instance

Figure 2 (a) shows an example of virtual network recovery using DVNR. Each virtual network V_i ($1 \leq i \leq 5$) has 3 virtual nodes, denoted as a_i, b_i, c_i. The solid line in the figure represents the substrate link, and the number on the substrate link represents the available bandwidth. The dotted line in the figure represents the virtual links have mapped on the substrate link currently, and the number on the virtual links represents the bandwidth requirement of the existing virtual links (For a better illustration of the example, only the bandwidth requirement of the virtual link $\overline{b_3c_3}$ is shown in the figure), and the number on the substrate links represents the remaining available bandwidth value after mapping an existing virtual link. Since there are five adjacent virtual networks share certain substrate resources (e.g. the virtual node a_2 of V_1 and the virtual node b_1 of V_2 share the substrate node C, and the virtual link $\overline{b_1c_1}$ of V_1 and the virtual link $\overline{a_2c_2}$ of V_5 share the substrate link $\overline{BC}$), the virtual machines of the five virtual networks can be abstracted into the topology shown in Fig. 2 (b). Assuming a substrate link $\overline{GF}$ fails unexpected, which will cause the virtual link $\overline{b_3c_3}$ mapped on $\overline{GF}$ to not work, resulting in the virtual network V_3 fail. After detecting this failure, the virtual machine of V_3 will first attempt to remap the virtual link $\overline{b_3c_3}$ with local network resources. Figure 2 (c) shows the substrate subnet that the virtual machine of V_3 can perceive. Obviously, the resources of the substrate subnet cannot meet the bandwidth requirements of the failed virtual link, and the virtual machine forwards the recovery request to the adjacent virtual machine. Figure 3 shows the process of virtual network recovery using the breadth-first forwarding strategy. The forwarding strategy will generate three forwarding paths $\overline{V_3V_5}$, $\overline{V_3V_2}$, $\overline{V_3V_4}$ at the same time. However, only the network bandwidth resources on the forwarding path $\overline{V_3V_4}$ meet the remapping requirement of the virtual link $\overline{b_3c_3}$. Therefore, the virtual machine of V_4 remapped the virtual link $\overline{b_3c_3}$ to the substrate link $\overline{GDEF}$ (the subnet merge order is shown in Fig. 4). After the virtual network V_3 is successfully restored, the virtual machine updates the substrate subnet topology information.

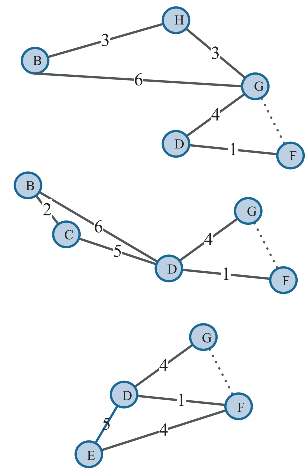

Fig. 4 Breadth-first subnet merge order

3. Experimental Details

The The substrate network topology was generated by the GT-ITM [9]. We chosen the set of parameters conforms with the ones used in the research literature [10]. There were 50 nodes and each pair of substrate nodes were randomly connected with probability 0.5. The capabilities of the substrate nodes and the bandwidth of the substrate links were uniformly distributed between 50 and 100. The probability that multiple substrate links fail within a period of time obeyed a Poisson distribution, and the average time interval between failures was set to 20 seconds.

Before the start of the simulation experiment, a total of 30 virtual networks with random topology in the network environment run on the shared substrate network. The capacity requirements of the virtual nodes and the bandwidth requirements of the virtual links were uniformly distributed between 0 and 50. The simulation experiment lasted for 2200 seconds to obtain stable experimental results.

We compared the performance of DVNR-2 (the number of message forwarding hop is less than or equal to 2), DVNR-3 (the number of message forwarding hop is less than or equal to 3), DVNR-4 (the number of message forwarding hop is less than or equal to 4) and the centralized virtual network recovery method CVNR that we implemented based on [5] in terms of recovery success rate and recovery efficiency.

4. Results and Analysis

As shown in Fig. 5, with the number of message forwarding hops increasing, the recovery success rate of DVNR is closer to that of the centralized VNR method. This is because the substrate network resources that DVNR can obtain are continuously expanded with the increasing of the number of message forwarding hops and the recovery success rate is correspondingly increased. If the DVNR does not limit the number of forwarding hops, the DVNR will eventually perceive the resources of the entire network, thereby being equivalent to the network resources that the centralized VNR method can perceive.

More importantly, the recovery efficiency of DVNR is significantly higher than that of the centralized VNR method. As shown in Fig. 6, the average recovery time of CVNR is about 2.4 seconds and the average recovery time of DVNR-4 is about 0.7 seconds. This means DVNR method can reduce the virtual network recovery time by about ∼70% on average. This is because virtual network recovery can be performed by DVNR in parallel, while the CVNR can only rely on serial recovery. We believe that this result is important for services or applications that have strict requirements for recovery time.

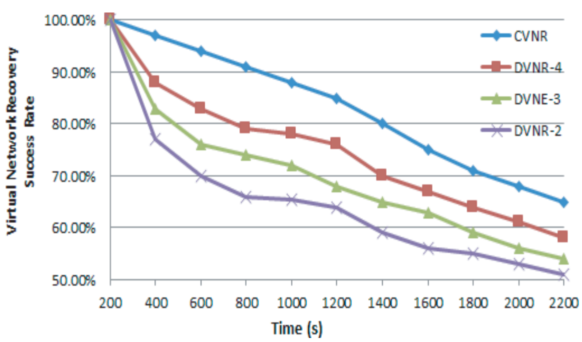

Fig. 5 Virtual network recovery success rate

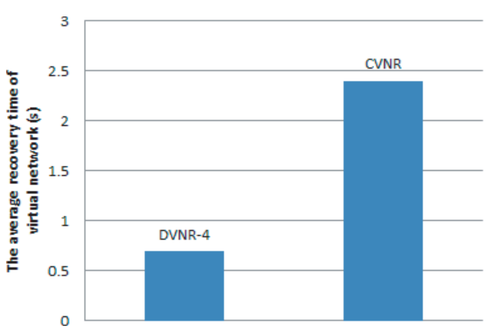

Fig. 6 Virtual network recovery efficiency

Acknowledgments

This work is support in part by National key research and development of China (No.2018YFB1701403), Natural science foundation of Hebei province (No.F2017210118), Colleges and universities in Hebei province science and technology research fund (No.QN2016270).

References

[1] N.M.M.K. Chowdhury and R. Boutaba, "A survey of network virtualization," Computer Networks, vol.54, no.5, pp.862–876, 2010.
[2] NSF Research: 'GENI,' http://www.geni.net/
[3] A. Fischer, J.F. Botero, M.T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," Commun. Surveys Tuts., vol.15, no.4, pp.1888–1906, 2013.
[4] S. Colman-Meixner, C. Develder, M. Tornatore, and B. Mukherjee, "A survey on resiliency techniques in cloud computing infrastructures and applications," Commun. Surveys Tuts., vol.18, no.3, pp.2244–2281, 2016.
[5] P. Zhang, H. Yao, M. Li, and Y. Liu, "Virtual network embedding based on modified genetic algorithm," Peer-to-Peer Networking and Applications, vol.12, no.2, pp.481–492, 2019.
[6] X. Siya, L. Peng, G. Shaoyong, and X. Qiu, "Fiber-wireless network virtual resource embedding method based on load balancing and priority," IEEE Access, vol.6, pp.33201–33215, 2018.
[7] N. Ogino, T. Kitahara, S. Arakawa, and M. Murata, "Virtual network embedding with multiple priority classes sharing substrate resources," Computer networks, vol.112(C), pp.52–66, 2017.
[8] M. Chowdhury, M.R. Riaihan, and R. Boutaba, "ViNEYard: Virtual network embedding algorithm with coordinated node and link mapping," IEEE/ACM Trans. Netw., vol.20, no.1, pp.206–219, 2012.
[9] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee, "How to model an internetwork," Proc. IEEE INFOCOM, San Francisco, March 2010:594-
[10] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," ACM SIGCOMM Computer Communication Review, vol.41, no.2, pp.39–47, 2011. 602.