

Fast Recovery from Link Failure in Software Defined Survivable Network

Guoqiang Yan^a, Ming Lei^b, Jieyong Zhang^c

Science and Technology on Information System Engineering Laboratory Nanjing, 210007, China

^aemail: misu.yan@qq.com, ^bemail: 001leiming@163.com, ^cdumu3110728@126.com

Keywords: survivable network; software defined networks; network simulation; survivable controller

Abstract. As the Internet becomes increasingly important to all aspects of society, it is critical to increase the survivability of the networks. A fast recovery algorithm was proposed by employing a centralized survivable controller to detect and recovery from link failures. A greedy strategy was used by relaying the separated link from a replaced path to improve recovery speed. The proposed algorithm is also adapted to node or area failures. Simulation experiments were conducted and showed that fast recovery algorithm has rapid recovery speed and it has low influence for up layer transport protocol.

Introduction

The increasing importance of the network information system has lead to network survivability becoming one of the critical feature on which almost every aspect of our lives depend. Network survivability in military refers to the ability of communication network tending to provide users with services after some facilities failed under the attack of enemy firepower. The difference between survivability and reliability is factors that lead to network failure is an external invasion, not a normal design can be expected. The difference between survivability and anti-interference is intrusion factor is usually in the form of physical damage [1].

Redundancy is the main way to solve the network survivability issues. To improve the redundancy of the network, firstly, survivability measurement model is established according invulnerability application requirements. Secondly, by adding nodes or links, computed network survivability ability can be improved in guide of the survivability model. The redundancy can be fully redundancy or shared redundancy. In fully redundancy, backup nodes and links was put into use only if a link or node fails. But in shared redundancy, all nodes and links is equally put into use. When there is a link failure, routing protocol or survivability mechanisms dynamically scheduled flows from bad routing to workable paths. With the increasing complexity of network environments and network applications increases, traditional backup technologies may emerge the following questions: 1) routing based redundancy method, since routers need to exchange routing table information, cannot switch to backup routings quickly in large scale networks. 2) Network topology based redundancy cannot distinguish the importance of network flows. So it cannot guarantee the critical flow recovery after network damage. 3) Routing information is distributed on every routing node, global network status information is hard to grasp, topology adjustment only partially carried out.

Software-defined network [2] (SDN) is a new network architecture proposed in recent years. It provides data plane and control plane separation. In data plane, switches forwarding and handling data packet according to flow tables; in control plane, a centralized controller connected with each switch through control link, controlled the flow entries in switches to direct data packet forwarding. In this paper a SDN based survivable network approach is proposed. In this approach, a survivable controller keeps detecting the network status in real-time. When network damage happens, controller quickly detects and computes recovery routing and flows to repair network to normal operation mode.

Related Works

Survivability is an important index to evaluate communication network performance. Early survivability research concentrated on network topology design and network capacity optimization. Network topology design means that when a number of the network nodes or links fail, the networks redundancy to meet the requirements of the degree of network connectivity. Capacity optimization problem means how to arrange a reasonable transmission path and the backup path for the communication services. Network topology design is the premise of solve the Capacity optimization problem [3].

Survivability index is the main tool to study the issue of survivability. Frank etc. first proposed two commonly used survivability indexes: connectivity and cohesion [4]. Later, Boesch and Wilkov proposed a more universal and practical connectivity and cohesion in [5]. In essence, connectivity and cohesion describe the network from the destruction of the degree of network. Survivability indexes from the perspective of network connectivity to describe and measure network survivability are: tenacity, integrity, discrete number and expansion coefficient. Another index describes network survivability by observing the process of removal of nodes or edges of invulnerability based on statistical physics parameters. commonly used measure have the number of communication piece, the largest piece scale communication, network diameter, the average path length, network efficiency [6].

Software-Defined Network Based Approach

In this section, a software-defined network survivable approach was conducted. The core design is survivable controller which can detect network failures and reorganized network routing paths. Design and implementation of survivable application based on Ryu controller [7] is also described.

Failure Detection.

Failure detection is the prior condition for a recovery process. Controller could detect a network failure in two ways: 1) *Active Report*, the switch will report to its controller events such as new port configured, port deleted or other events which hints a link failure; 2) *Passive Report*, a switch damage may cause its neighbor switch to report port status change events. Table 1 shows various events monitored by the Ryu controller. Network failures can be primarily divided to two categories, one is the link damage, corresponding to the *EventPortStatusChange* event; the other one is node damage, corresponding to *EventDpLeave* event. After these events were detected, the controller started a recovery process.

Table 1. Failure events Ryu controller detect

Event name	Explanation	Parameter
EventDpEnter	a switch registered	dpid
EventDpLeave	a switch unregistered	dpid
EventPortAdd	add a port	dpid, port_num
EventPortDelete	delete a port	dpid, port_num
EventPortModify	modify a port	dpid, port_num
EventPortStatusChange	port status change	dpid, port_num

Fast Recovery Algorithm.

In order to recovery from a failure as fast as possible, fast recovery algorithm only repair routes separated by damaged links. For node failure situation, fast recovery algorithm will repair all routes pass through the broken node.

Suppose the link between node a and node b named (a, b) separated by a damage, fast recovery algorithm first finds all routes pass through (a, b) . For example, a route named $r1$ in node a to destination $d1$ whose next-hop is node b may suffer from this failure. Usually, there is an opposite route named $r2$ in node b to destination $d2$ whose next-hop is node a need to be repaired.

FastRecover function in Fig. 1 is the main recovery function; it returns a route list which

contains *dpid*, *destination* and *nexthop* address. Four input parameters *dp1*, *dp2*, *ip1*, *ip2* means to recovery link between *dp1* and *dp2* whose ip address respectively are *ip1*, and *ip2*. In *FastRecover* function, *all_pairs_shortest_path* return the shortest path matrix of global topology graph *G*. The *FindRoutesToUpdate* function will traverse new shortest path from *dp1* to *dp2* to find the right node to update the flow entry.

algorithm 1 Fast Recover Algorithm

Input: *G dp1 dp2 ip1 ip2*
Output: routes

```

function FINDROUTESToUpdate(routes, dpid, path)
2:   ret = [ ]
   for r in routes do
4:     for i = 0 → length(path) - 1 do
       node1 ← path[i : i + 2][0]
6:       node2 ← path[i : i + 2][1]
       addr1 ← get_nexthopaddr(node1, node2)
8:       addr2 ← get_nexthopaddr(node2, node1)
       reverse_gw ←
10:        get_gateway_by_dest(node2, r['dest'])
       if reverse_gw ≠ addr1 then
12:         ret.append(node1, r['destination'],
           node2, addr2)
14:         break
       end if
16:     end for
   end for
18:   return ret
end function
20: function FASTRECOVER(dp1, dp2, ip1, ip2)
   r1 ← routes from dp2 with nexthop from dp1
22:   r2 ← routes from dp1 with nexthop from dp2
   G.remove(dp1, dp2)
24:   P1 ← all_pairs_shortest_path(G)[dp1][dp2]
   P2 ← all_pairs_shortest_path(G)[dp2][dp1]
26:   routes = [ ]
   routes.append(FindRoutesToUpdate(r1, dp1, P1))
28:   routes.append(FindRoutesToUpdate(r2, dp2, P2))
   return routes
30: end function

```

Fig.1. Pseudo-code of fast recovery algorithm

Simulation and Results

In order to demonstrate the efficiency and evaluate influence for the up-layer protocols of the fast recovery algorithm, we integrated software defined network simulation function into the common open research emulator (CORE)[8]. Circle topology and grid topology were choozen to be the redundant topology.

Simulation Settings.

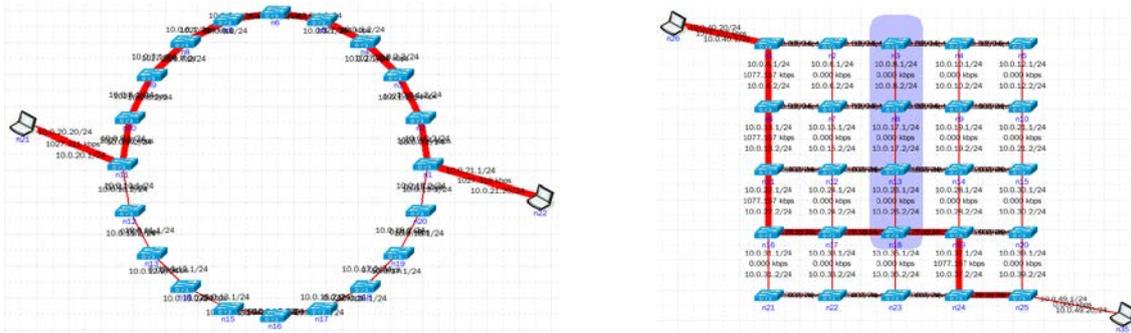
CORE is a large scale network emulator based on container virtualization. For containers lightweight kernel virtualization, CORE can emulate a large number of routers which has own namespace and own network protocol stack. Container provides the same computation environment as the host. All applications can run in container without any modification. A container run routing protocols could be a virtual router, a container run server applications could be a virtual sever. Container can also send and receive packets if it is connected to a peripheral network card.

In order to build software defined network simulation environment, we integrated open vswitch bridge [9] as the second layer equipment to CORE. In front configuration side, a new OVS type of node was added to allow user to configure open vswitch bridge network. In background side, we implemented OVSKernelNet class which inherits from PyCoreNet class to process OVS type node messages. In the startup function, ovs-vsctl command will be executed to build open vswitch

bridges.

Topology Configurations.

Topology survivability is the precondition for a damaged recovery. It means there are at least more than 2 different paths from source to destination. We choose two representative topologies: circle and grid for the experiments. For circle topology, there are two paths between every two pair nodes. For grid topology, there are more than two paths between every two pair nodes.



(a) The circle topology

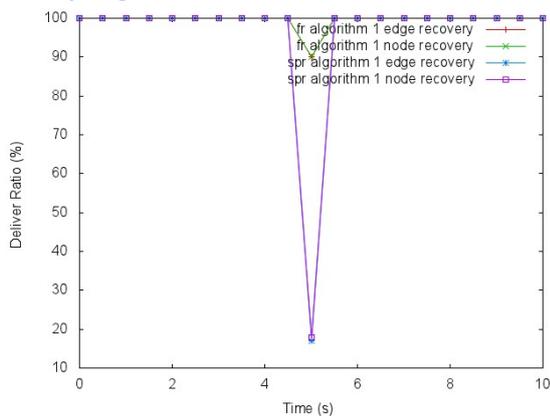
(b) The grid topology

Fig.2. circle and grid topologies

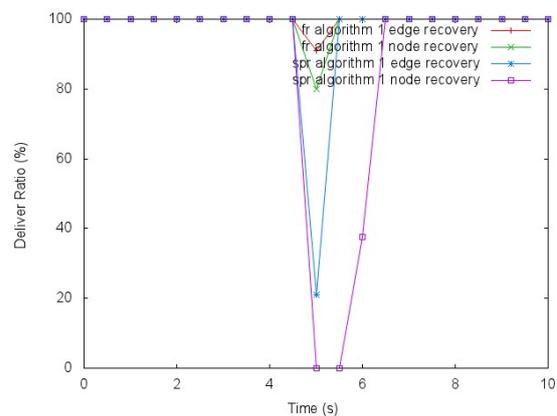
As fig. 2 shows, besides switches, two virtual PCs were configured in the network as source and destination. Iperf tool[10] is employed to generate network flows which can be observed from CORE. Both UDP and TCP test were conducted. In UDP test, all packets will be counted by sequence number field in IP packet in the destination side. In TCP test, the throughput could be measured to evaluated influence on up layer protocol of recovery algorithm.

Result and Analysis.

Figure 3(a) shows the UDP deliver ratio of fast recovery algorithm and shortest path recovery algorithm for single node damage and single link damage under 20 nodes circle topology. Peer to peer UDP bandwidth was 1 Mbps, and report interval was 0.5 seconds which means in every 0.5 second 45 packages was delivered. As figure shows, both two algorithms recovery deliver ratio to 100 percentages from damage in 0.5 second. The shortest path recovery algorithm in single link and single node damage restoration package loss were 37 and 38, about 17.7% and 15.5% transmission ratio. The fastest recovery algorithm in single link and single node damage restoration package loss were 5, about transmission rate of 90%. We can estimate the shortest path algorithm and the fastest recovery algorithm restoration time is about 180ms and 60ms.



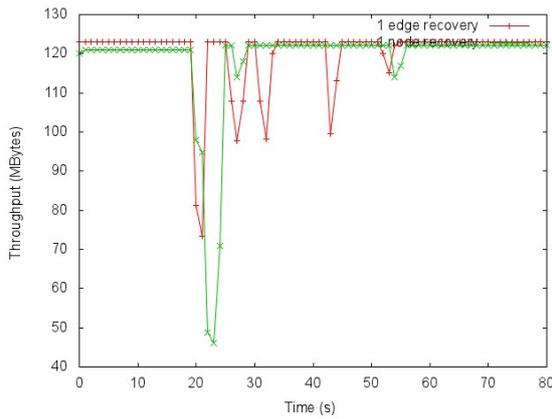
(a) UDP delivery ratio under circle topology



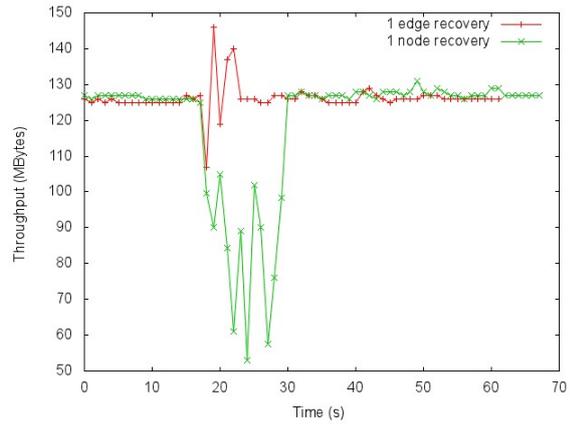
(b) UDP delivery ratio under grid topology

Fig.3. UDP delivery ratio in recovery experiment

Figure 3(b) shows the UDP deliver ratio of fast recovery algorithm and shortest path recovery algorithm for single node damage and single link damage under 25 nodes grid topology. 1Mbps bandwidth and 0.5 second report interval were set. As figure shows fast recovery algorithm still recovery in 0.5 second with package loss of 5 and 10 for single link damage and single node damage but shortest path recovery algorithm in 1.5 seconds with package loss of 67 and 146.



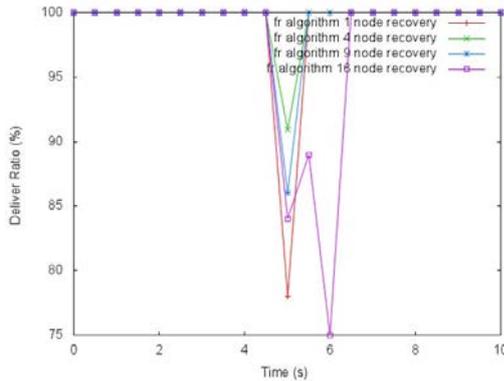
(a) TCP throughput under circle topology



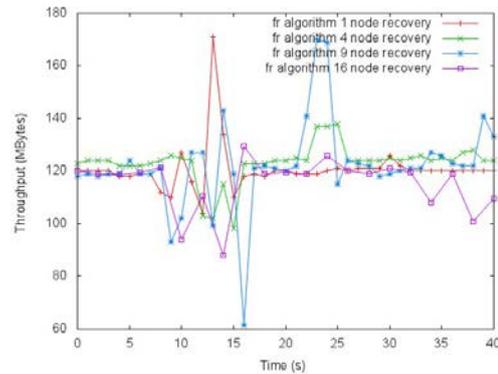
(b) TCP throughput under grid topology

Fig.4. TCP throughput in recovery experiment

TCP throughput results were plotted in fig.4(a) and fig.4(b) for circle and grid topologies. As figure shows, peer to peer throughput could climb up to about 120MBytes, but in damage recovery process, throughput declined to 73.3MBytes and 46.23MBytes. In fig.4 (b) throughput in recovery process could be larger than normal state for the reason of TCPs retransmission. Similar to UDP test, single link recovery performance is better than single node recovery.



(a) Area failure recovery under circle topology



(b) Area failure recovery under grid topology

Fig.5. Area failure recovery experiment

Area failure recovery results were plotted in fig.5 (a) and fig.5 (b) for grid topologies. In each simulation we choose 1, 4, 9 and 16 nodes for failure area. Fig.5 (a) shows UDP deliver ratio degradation in recovery process. Larger failure area results in a longer recovery time because there are more broken links needs to be repaired. Fig.5 (b) also shows that area damage will not cause a totally disconnection between sender and receiver. The lowest throughput will not be less than 60 MBytes.

Conclusion

Survivability is an essential property of the Future Internet, although it has been an active area of research for a half-century. Software defined networks and virtual networks technologies introduce a fresh new thought to design and recovery survivable networks. This paper aims to employ survivable application in Ryu controller to monitor network damages and help switches to recovery. Simulation results show that the proposed algorithm has fast and strong recovery ability. Even in TCP recovery test, single link or single node damage would not have serious influences on upper-layer application.

Acknowledgement

In this paper, the research was sponsored by the Chinese National Science Foundation Program (Project No. 71571186).

References

- [1] D. Alderson, L. Li, W. Willinger, and J. C. Doyle. Understanding Internet Topology: Principles, Models, and Validation [J]. IEEE/ACM Transactions on Networking, 2005, 13(6): 1205-1218,
- [2] McKeown N. Software-Defined networking[C]. In: Proc. of the INFOCOM Key Note. 2009. <http://infocom2009.ieeeinfocom.org/technicalProgram.htm>
- [3] L. Fratta, U. G. Montanari. A recursive method based on case analysis for computing network terminal reliability[J]. IEEE Trans. On Communication, 1978, 26 (8): 1166-1177
- [4] Watts D J, Strogatz S. Collective dynamics of 'Small-World' networks[J]. Nature. 1998,393:440-442
- [5] Wilkov R. S. Analysis and design of reliable computer networks[J]. IEEE Transactions on Communications, 1972, 20(3):660-678
- [6] Barabasi A-L, Albert R. Emergence of scaling in random networks[J]. Science. 1999,286:509-512
- [7] M. Monaco, O. Michel, E. Keller. Applying operating system principles to SDN controller design[C]. In: Proc. of the ACM Workshop on HotNets. 2013. 1-7.
- [8] J. Ahrenholz, C. Dnailov, T. R. Henderson, J. H. Kim. CORE: A real-time network emulator[C]. in IEEE MILCOM Conference, Nov. 2008,1-7.
- [9] The Stanford OpenFlow Team. OpenFlow Switch Specification Version 1.0.0 [S], 2009, <http://www.OpenFlow.Org/documents/OpenFlow-spec-v1.0.0>.
- [10] Tirumala Ajay. Iperf: The TCP/UDP bandwidth measurement tool[OL]. (2011.5) [2015-10-20]. <https://iperf.fr>.