Advances in Intelligent Systems Research, volume 153
International Conference on Applied Mathematics, Modeling and Simulation (AMMS 2017)

Lossless Adjustment in Software Defined Survivable Network:A Simulation Study

Guoqiang Yan*, Xin Huang, and Jieyong Zhang
Science and Technology on Information System Engineering Laboratory, Nanjing, 210007, China
*Corresponding author

***Abstract*—As the Internet becomes increasingly important to all aspects of society, it is critical to increase the survivability of the networks. A software-defined network based disruption recovery approach is proposed. It employs a centralized survivable controller to detect damage to network such as link disconnection or node broken, then it recomputed the transmit path of every influencing routing. In the first fastest recovery phase, local recovery routing would be investigated between damaged links and areas; in the second lossless adjustment phase, shortest routing path would be searched to replace local recovery routing in a lossless manner. Simulation experiments were demonstrated and showed that software-defined network based disruption recovery has a more rapidly recovery ability and has low influence on up layer transport protocol.**

Keywords-survivable network; software defined networks; network simulation; lossless adjustment

I. INTRODUCTION

The increasing importance of the network information system has led to network survivability becoming one of the critical feature on which almost every aspect of our lives depended. Network survivability in military refers to the ability of communication network tending to provide users with services after some facilities failed under 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 the 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 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 were put into use only if a link or node failed. But in shared redundancy, all nodes and links were equally put into use. When there is a link failure, routing protocol or survivability mechanisms dynamically scheduled flows from bad paths to workable paths. With the increasing complexity of network environments and network applications increase, 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 damaged. 3) Routing information is distributed over every routing node, global network status information is hard to collect, and topology adjustment only partially was 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 forward and hand data packets according to flow tables; in control plane, a centralized controller which connects with all switches through control link, controls the flow entries in switches to direct data packet forwarding.

In this paper a two-phase survivable network approach based on software defined network is proposed. Based on the network topology redundancy design, a invulnerability controller keeps detecting the network damage in real-time. When network damage happens, 1) fastest recovery, local recovery routing would be investigated between damaged links or areas; 2) lossless adjustment, shortest path routing would be conducted to replace local recovery routing in a lossless manner. Simulation experiments were demonstrated and showed that the proposed approach has a more rapidly recovery ability and has low influence for up layer transport protocol.

II. 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 failed, 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 solving 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 measurement have the number of

communication piece, the largest piece scale communication, network diameter, the average path length, network efficiency [6].

III. SOFTWARE-DEFINED NETWORK BASED APPROACH

In this section, a software-defined network survivable approach was proposed. 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 conducted.

A. 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 hint 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 into 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 I. 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

B. Fast Recovery

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

As Figure. I shows, there exists a short path from n1->n2->n3->n4.When n2->n3 damaged, fast recovery algorithm will find a new path from n2 to n3, and only update flows in node n2 and n3. Fewer nodes affected are the reason why it is time-efficient.

Figure. II shows 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 address respectively are ip1, and ip2. In FastRecover function, all_ pairs_ shortest_ path returns 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.

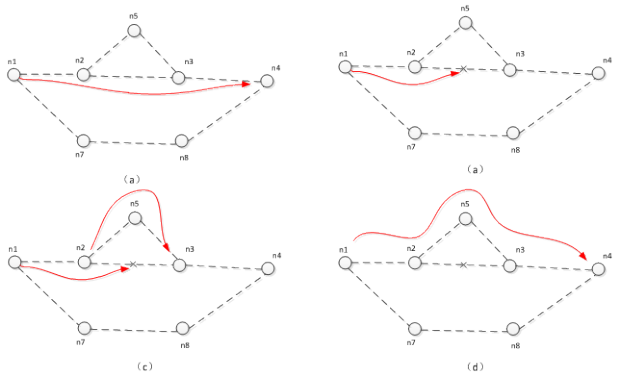


FIGURE I. SKETCH MAP OF FAST RECOVERY

algorithm 1 Fast Recover Algorithm
Input: G $dp1$ $dp2$ $ip1$ $ip2$
Output: routes
function FINDROUTESTOUPDATE($routes, dpid, path$)
  $ret = [\,]$
  for r in $routes$ **do**
    for $i = 0 \rightarrow length(path) - 1$ **do**
      $node1 \leftarrow path[i : i + 2][0]$
      $node2 \leftarrow path[i : i + 2][1]$
      $addr1 \leftarrow get\_nexthopaddr(node1, node2)$
      $addr2 \leftarrow get\_nexthopaddr(node2, node1)$
      $reverse\_gw \leftarrow$
        $get\_gateway\_by\_dest(node2, r['dest'])$
      if $reverse\_gw! = addr1$ **then**
        $ret.append(node1, r['destination'],$
          $node2, addr2)$
        $break$
      end if
    end for
  end for
  $return\ ret$
end function
function FASTRECOVER($dp1, dp2$, $ip1$,$ip2$)
  $r1 \leftarrow routes\ from\ dp2\ with\ nexthop\ from\ dp1$
  $r2 \leftarrow routes\ from\ dp1\ with\ nexthop\ from\ dp2$
  $G.remove(dp1, dp2)$
  $P1 \leftarrow all\_pairs\_shortest\_path(G)[dp1][dp2]$
  $P2 \leftarrow all\_pairs\_shortest\_path(G)[dp2][dp1]$
  $routes = [\,]$
  $routes.append(FindRoutesToUpdate(r1, dp1, P1))$
  $routes.append(FindRoutesToUpdate(r2, dp2, P2))$
  $return\ routes$
end function

FIGURE II. PSEUDO-CODE OF FAST RECOVERY

C. Lossless Adjustment

As Figure. III shows, path n1->n2->n5->n3->n4 is the recovered path by a Fast Recovery algorithm. But there exists a short path n1->n7->n8->n4. Lossless Adjustment can solve this problem. It finds a better shorter path, and builds flow reversely. As Figure. III (b) shows, n8->n4 flow is updated firstly. Then n7->n8. Finally, n1->n7 is updated. It claimed that in last flow updating period, a priority flow technical will be employed. It makes sure flow will not be interrupted. As Figure. III (f) shows, after all the shortest path flows are updated; it will switch to new path by deleting flow in the first node n1.

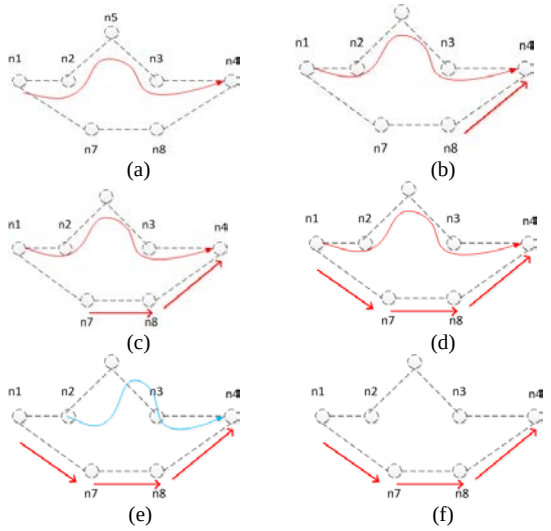


FIGURE III. SKETCH MAP OF LOSSLESS ADJUSTMENT

Figure. IV shows lossless adjustment pseudo-code. It returns a route list which contains dpid, destination and nexthop address. Three input parameters dp1, dp2, routes means to recovery links between dp1 and dp2, and routes to be recovered . In LossLessAjustment function, paths which need to recover are investigated; then build new flow along with short path reversely; third after a new flow is switched, old flow will be deleted by this function.

Algorithm 2 LosslessAjustment

Input: G dp1 dp2
Output: routes

```
function LosslessAjustment(dp1, dp2, routes)
    for r ∈ routes which pass through dp1 and dp2 do
        a ← r.start; b ← r.end
        left ← Length(routes[a][dp1])
        right ← Length(routes[dp2][b])
        middle ← Length(routes[dp1][dp2])
        SP ← all_pairs_shortest_path(G)[a][b]
        s ← length(SP)
        if s < (left + middle + right) then
            for n ∈ SP.reverse() do
                new_route = (dest = r.dest, next = n's next in SP)
                if n ∈ r then
                    add new_route in n with low priority
                else
                    add new_route in n
                end if
                routes.append(new_route)
            end for
            for n ∈ r do
                delete route(dest = r.dest, next = ns next in r) in n;
            end for
        end if
    end for
    return routes
end function
```

FIGURE IV. PSEUDO-CODE OF LOSSLESS ADJUSTMENT

IV. SIMULATION AND RESULTS

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

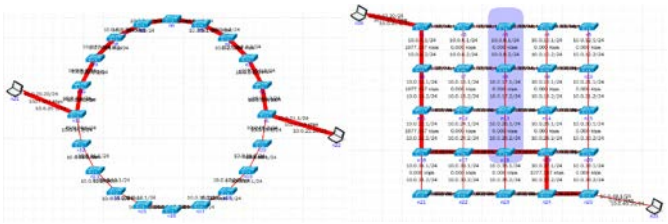
A. Simulation Settings

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 have 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 runs routing protocols could be a virtual router, a container runs 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 sides, a new OVS type of node was added to allow user to configure open vswitch bridge network. In the background side, we implemented OVSKernelNet class which inherits from PyCoreNet class to process OVS type node messages. In the startup function, a ovs-vsctl command will be executed to build open vswitch bridges.

B. 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) circle topology (b) grid topology

FIGURE V. CIRCLE AND GRID TOPOLOGIES

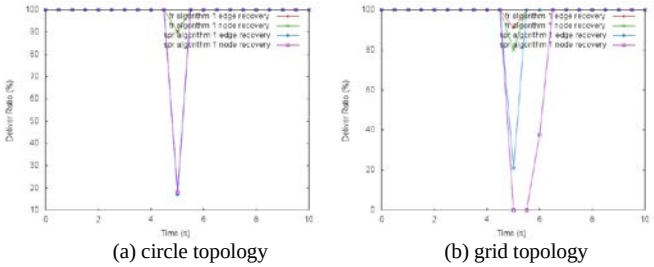
As Figure. V shows, besides switches, two virtual PCs were configured in the network as source and destination. Iperf tool [10] is employed to generate network flows which can be observed from CORE. Both UDP and TCP test were conducted. In UDP test, all packets will be counted by sequence number field 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.

C. Simulation Results

The simulation experiment mainly examines the efficiency of the recovery algorithms, mainly for single node damage, single link damage analysis and testing. Shortest Path First Algorithm (spr) is single-stage algorithm which investigates the shortest path immediately when damage occurred. Fast recovery with lossless adjustment is the proposed two-stage algorithm.

Figure. VI (a) shows the UDP deliver ratio of fast recovery algorithm and shortest path recovery algorithm for single node

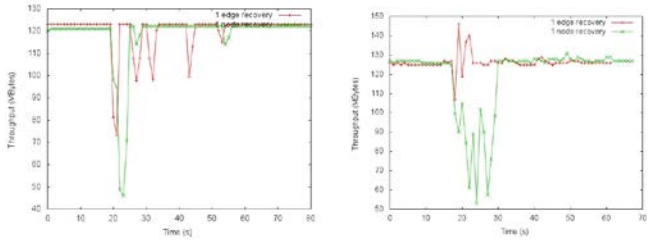
damage and single link damage under 20 nodes circle topology. Peer to peer UDP bandwidth was 1 Mbps, and report interval was 0.5 seconds which means in every 0.5 second 45 packages was delivered. As figure shows, both two algorithms recovery deliver ratio to 100 percentages from damage in 0.5 second. 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) circle topology (b) grid topology

FIGURE VI. UDP DELIVERY RATIO

Figure. VI (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.

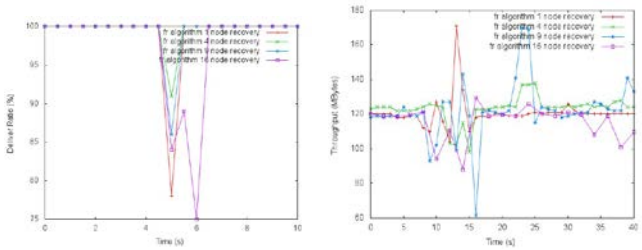


(a) circle topology (b) grid topology

FIGURE VII. TCP THROUGHPUT

TCP throughput results were plotted in Figure. VII (a) and Figure. VII (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 Figure. VII (b) throughput in recovery processed 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.

Area failure recovery results were plotted in Figure. VIII (a) and Figure. VIII (b) for grid topologies. In each simulation we choose 1, 4, 9 and 16 nodes for failure area. Figure. VIII (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. Figure. VIII (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.



(a) circle topology (b) grid topology

FIGURE VIII. AREA FAILURE RECOVERY

V. 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.

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 metworking[C]. In: Proc. of the INFOCOM Key Note. 2009. http://infocom2009.ieeeinfocom.org/technicalProgram.htm

[3] L. Fratta, U. G. Montanari. Arecurisive method based on case analysis for computing network terminal reliability[J]. IEEETrans. 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.l999,286:509-512

[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.QpenFlow.Org/documents/QpenFlow-spec-vl.0.0

[10] Tirumala Ajay. Iperf: The TCP/UDP bandwidth measurement tool[OL]. (2011.5) [2015-10-20]. https://iperf.fr