Resilience of the Internet Protocol by Improving the Recovery Mechanism via PING Agent

Radwan S. Abujassar

***Abstract*—The recovery algorithms in the network lead to reduce deliver data time between source and destination by interoperate between layer 2 and layer 3. We created three topologies and each topology has a different number of links and nodes. The algorithm mechanism is to make the source node re-route the packets when failure occurs in a new path through an adjacent node. Hence, the source node will send the packet to one of the adjacent nodes, which has chosen randomly. This node will use a different path, which does not affected by the failure of the destination node. In addition, is not necessary that this secondary path be the shortest to make recovery time faster. (When the source node re-routes the packet directly to the adjacent node the routing protocol in the source node will update the routing table and re-compute the secondary shortest path.**

***Index Terms*—Internet protocol, Ping agent, open shortest path first (OSPF).**

I. Introduction

Many papers have recently published about recovery in the network. Each one mentioned different techniques and different algorithms to improve the recovery time and fast convergence without any modifications to the network such as hardware, protocols. The routers are responsible for receiving and forwarding the data through an interconnected a set of networks. In addition, the router with lower layers must avoid that portion of network that has failed or congested. Many protocols help the router to take decisions to select the best route, which is the shortest one to transmit the data. Recently, many algorithms have published to make the computation of the shortest path faster by reducing the waiting time for calculating the new shortest path. The waiting time is to confirm that a failure has occurred. This is because the instability of links will make the algorithm work frequently, which is degrading the network. The current algorithms are waiting 5 sec to be sure the failure has occurred [1]. This waiting time raises two issues: firstly, the recovery time will not be desirable and secondly the buffer size may not have enough space to retain the packet until the new shortest path has computed. In our algorithm, the QoS forms one of many factors, we must consider it to make the network more efficient with high service for all users. Many papers have published about QoS. They indicate how the new architectures have developed to support many applications such as video, VOIP [2]. The QoS demand constrains must be achieved to find a path can be tolerant any traffic without degrade the network. In the network layer (layer3), the mechanism of routing protocols is to build the routing table, compute the best path between source and destination, and additionally it can detect failure through “Hello Packets” in OSPF protocol. However, the routing protocol use one of network characterizes to compute a primary path such as metric (cost), hop-count, delay or bandwidth. In addition, the recovery techniques nominate a candidate paths based on the cost metric.

Manuscript received December 23, 2015; revised May 3, 2016.

Radwan S. Abujasar is with the ITC program, Arab Open University, Kuwait, Alardiya (e-mail: r.abujassar@aou.edu.kw).

II. Related Works

Efficient routing protocol algorithms have built for achieving the robustness and fast convergence within a short time, in case failure. In [3], the author indicates about the cost of links in the network and traffic engineering. As we mention above, the links cost consider one of the important parameters to determine the best path through the routing protocol algorithm. The OSPF protocol based on the dijkstra algorithm to compute the shortest path; the minimum path cost will determine by compare it with other candidate path. The packet will re-route from the backup path through the routing protocol, the problem here when the backup path pass other traffic then the load will become high and the congestion lead to drop the packet. Therefore, the traffic engineer is coming to solve this problem by allocate the traffic through the equal cost path with less utilization.

The OSPF optimize multiple path protocol OSPF (OMP) emphasis to solve the load on the path [4], and achieve optimal distribution load balance in the network in case failure. However, there are two drawbacks of OMP mechanism. First, it needs more memory size to store vectors. Second: the information will generate without any deterministic and unpredictable. However, if we assume there is more than one path has the similar cost with primary shortest one, then we can shift and divide the traffic from all ECMP to decrees the utilization on the primary path, and the load balancing will be achieved. In addition, the ECMP will avoid the loop in the network [5].

Open Shortest Path is an extremely important protocol that is used in the network. The OSPF protocol uses a Dijkstra algorithm, which computes the shortest path between source and destination in a short period time. It is described in detail later.

The OSPF protocol performs a number of internal tasks: firstly, processing the LSAs. Secondly, performing SPF and updating the forwarding information and flooding the LSAs. OSPF [6] is a links state routing protocol and it has a correlation with the delay, which is associated with the number of routers and links.

There are many factors leading to delay in the network as follow:

1) LSA flooding
2) Copying LSU
3) The delay involved in computing the shortest path by using Dijkstra's algorithm.

By the network operator, each link will assign a weight and the shortest path is computed by using these weights as lengths of links [7]. The highlight points are, if the router has multiple outgoing links then one of them will be a primary path, which is the shortest path to the destination, and the other outgoing links may configured as a backup links that can be used in case failure. In this case, the router will use a hash function with some information in the header to assign packets for which outgoing links will transit to the destination. Hash functions well-defined a procedure or function that is used to speed table lookup to find item in a database. In addition, Hash functions are used in hash tables to locate quickly a data record [8], [9].

In our mechanism, we measure the load balancing has measured by:

Load Link Metric = (traffic size)/(link capacity × Time period); Link Cost (utilization) =link cost × w × utilization

There are two kinds of the dijkstra algorithm; first: dijkstra algorithm to compute the best path by removing the links with bandwidth less than the threshold. Second, on demand dijkstra algorithm, which is generate the shortest path tree to the pre-computation mode, and depend on the bandwidth request the node will be added in the tree [10], [11].

The IP recovery emphasize on two cases: first, the time to detect the failure. Second the time to compute the shortest path. Hence, we will discuss about some techniques that contribute to improve the recovery time by achieving a desirable result. Failure Insensitive technique is one of the efficient techniques in the IP recovery. When the failure occurs, FIR mechanism will inform the source node about failure through encapsulate the packet (encapsulate in encapsulate) with a special header and return back the packet to the source, the source node will notice about the failure through the new header, then will send packet to another path, which is disjoint with the primary path. FIR mechanism will avoid loop in the network, but the recovery time will be not desirable. The drawback in this technique, when the packet will encapsulate two times that will lead to consume the bandwidth, delay and congestion in the network [12]. On the other hand, IPFRR is an applicable technique, it includes the LFA, U-turn and not-via address [13]. The draw back in IPFRR technique is the loop free not guaranteed. In addition, not-via address need to encapsulate/de-capsulate, which effects on network performance as we indicate above.

III. Our Technique

A. Network Simulation (NS2)

NS2 is an open source event-driven simulator designed for research in computer communication networks. It is an object-oriented simulator written in C++ with an OTcL interpreter as the front end. In addition, it contains modules for numerous network components, such as, routing, transport layer and application. NS2 contributes towards investigating network performance. In addition, NS2 has easy to use scripting language to configure networks and observe the results it generates. NS2 has become the most widely used, and it is open source network simulator. NS2 is free software and we can download it from the internet.

B. Experiment Methodology

There is a huge amount of data transmission through networks, and these data can send from source to destination through more than one route. In, addition, there are many routing protocols can be used in the network such as, RIP, EIRGP and OSPF. In this experiment we will use OSPF protocol for many reasons as discussed above. Hence, OSPF use the routing algorithm (Dijkstra's algorithm) to compute the shortest path between source and destination depending on the metric. The metric can be measured on different ways such as the number of hops, delay, bandwidth or cost.

TABEL I: Parameter Values

Names	Parameter values
Packet size	1000 byte
Packet interval	0.01 ms
Simulation time	10s
Number of nodes	8
Link Capacity	1Mb
Type of Queue	RED
Routing protocol	Link State
Link Down & UP	Randomly/Uniform
Rate	500 Byte
No. of Links	13

In this experiment, we configured the Link State protocol on all nodes, which is currently available in NS2 simulator. In addition, this topology contains thirteen links and seven nodes. The propagation delay for each link is configured 2ms, and the capacity for all links between (1 and 2) MB, see Table I. In addition, all CBR traffics use UDP protocol in layer4 to transmission the traffic from source to the destination. In addition, there are three CBR traffics (Traffic-1, Traffic-2, and Traffic-3); each one has a different flow number (0, 1, 2) respectively. We have created twelve variables as a [Random/Uniform] exponential to make failure occurs randomly, and make traffic start and stop randomly during the run simulation time. Rather than, when failure occurs on primary paths, the LS send LSAs after detect failure to update the routing table and compute the new shortest path route, during that there are many problems may occurs such as congestion, loop and loss packets. This experiment aims to show how the packets will drop and the generation delay become high through many factors we will discuss them through the result. The congestion model used in this experiment tries to emulate the behaviour of Random Early Drop (RED) [13], which is usable for congestion avoidance in packet-switched networks. However, when the average queue size exceed a particular limitation the packet will drop probability (P) at a router interfaces. Hence, the main idea of RED queue keeps the average queue size under this limitation. In addition, when the RED queue increases linearly from a value 0 to max_p at the average buffer occupancy queue length, if the queue length exceeds 1, in this case all the incoming packets are dropped. RED queue is more useful when we use TCP protocol for two reasons as follow: First, when the average size for RED queue become over 75% it should send notification to decrease the window size, then that will avoid congestion and loss packets in the network. Second, the TCP detects the congestion after the packet dropped.

This topology will assist in showing how the OSPF protocol works when the failure occurs.

C. Ping Agent

It is clear that the OSPF protocol can detect failure through “Hello packets”. In [13], Hello packets will take long time to detect failure. Therefore, if we want to change the time for Hello packets that will lead to different problems such as increase load on the link, consume bandwidth and propagation delay. Hello packets send every 10 sec, the failure will confirm after four hello packets send without any acknowledgment that means after 40sec. Hence, there are two basics work for the hello packets. First: detect failure. Second, establish and maintain neighbour relationships The author made some experiments to send hello packets in less than 10 sec, and then check in which time consider the best to send hello packets for detecting the failure faster. The hello packets size will increase depend on the neighbour numbers. On the other hand, header packet for ping agent includes two fields:

1-Ret

2-Time interval

The char 'ret' is going to be set to '0' if the packet is on its way from the sender to the node which is being pinged, while it is going to be set to '1' on its way back. The double 'send time' is a time stamp that is set on the packet when it has sent. In this example, we will prove the ping agent will not make any load that degrading from the network performance on each link on the path between source and destination.

Example: Assume there are two nodes {A, B} the source will be A and B the destination, the link capacity 1Mb and the packet size =1000kb. We will configure the LS protocol work on this topology and the rate =500 kb.

We will compute the utilization of link with hello packet and ping packet and we can see the different between them.

$$\text{Link utilization } \% = (\text{data bits } \times 100) / (\text{bandwidth } \times \text{interval})$$

Hello packets size = 384bit because we have just one neighbour; Ping packet=16 bit

We compute first the hello packet with assume the interval time 10sec.

$$\text{Link utilization}\% = (384 \times 100)/(1\text{Mb} \times 10000\text{ms}) = 0.036621 \times 10\text{-}4\%$$

For ping agent:

$$\text{Link utilization}\% = (16 \times 100)/(1\text{Mb} \times 10000\text{ms}) = 0.0015258 \times 10\text{-}4\ \%$$

As we can see the ping, agent size is negligible even if we make the time interval less than 10sec. In case, the link utilization =100% that will avoid the ping agent and hello packets to send during that until the utilization go less than 100%. In this case, the hello packet and ping packet will wait in the queue. If we decrease, the time interval for both the utilization of link will be as follow:

The total link utilization with hello packet is

$$\% = [((1000 \times 8)+384) \times 100]/1\text{Mb} \times 5000\text{ms} = 0.0015973$$

The total link utilization with ping agent is

$$\% = [((1000 \times 8)+16) \times 100]/1\text{Mb} \times 5000\text{ms} = 0.0015272$$

IV. Result

In this experiment, we ran 10.0s simulations ten times, and configured the LS protocol to work on this topology. The CBR traffics for all sources nodes started to send from 0.5s to 9.0s. During that time we caused many failures in different links. Firstly, we caused three links to become inoperative. The failures occurred at different times because we have created six variables [Random/Uniform]. The first three variables have min_v equal to 0.5s and the max_v equal to 5.0s. The link went down between these times. The remaining variables have min_v equal to 6.0s and max_v equal to 9.0s. These values indicate the time for the links go up.That means that the failures it will not remain for a long time, and through these results, we can measure their affect.

We have created a small topology, and we use UDP agent with Exponential traffic between the source and destination. The capacity of link 1Mb and the run simulation time is 10 sec. Through the trace file, we compute the utilization of link with awk program, and then we have produced the result as above. The load is decrease, when the time for sending the hello packet and ping agent increased. However, the load is high when we are sending hello and ping packets at 0.5sec and the load is stable for the ping agent on different times. In the Fig. 1 below, we will see the hello and ping packet. We make small experiment to prove when the ping agent send frequently with less than 10 sec the load will be less than hello packets. The idea here, we will use the ping agent packet to send frequently between source and destination, and we will keep the hello packet sending as typically. Hence, failure can be detected faster than hello packets. Therefore, when we speak about recovery time within 50 ms, then the failure should be detected in less than that time.

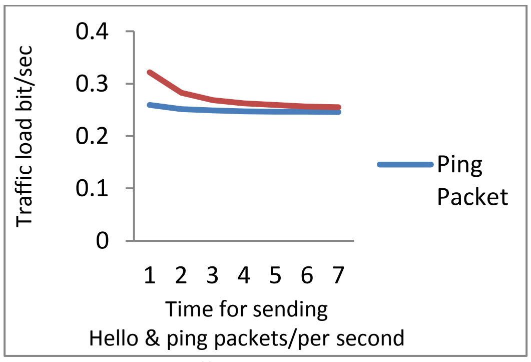

Fig. 1. UDP traffic with Hello & Ping packets.

A. Number of Failures

When the number of links down the protocol will take more time to re-compute and re-route the packets through another path. We can invest layer2 here to detect failure faster. In Layer2 the router can determined the destination through MAC address, so when the failure occur the MAC address will remove from the IP header. By Logical Layer control, the higher layer can interact with lower layers. In addition, the loss signal and light will indicate link or node down, then logical link layer (LLC) will inform layer 3 about the failure. Fig. 2 illustrates the design of a scenario using a small topology. This topology acts a simple example to show how the time recovery will increase when failure span to different link. Each node has at least two egresses that can re-route packets through either one of them when failure occurs.

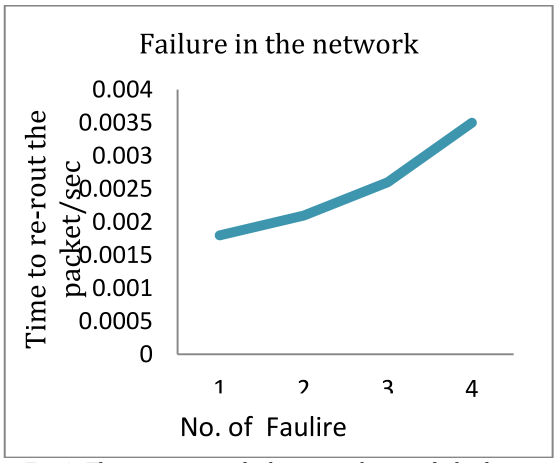

Fig. 2. The time increased when more than one links down.

V. Conclusion and Future Work

As we know the OSPF protocol compute the shortest path depend on the cost, which is available currently in NS2 simulator. Our technique is to make the dijkstra algorithm compute the best path through two parameters. First: compute the best path depend on the available bandwidth. Second: compute the best path depend on the delay of links. The aim of this technique, when the failure occurs we will assume the second best path is passing traffic; therefore to avoid loss of packets we will make the alternative path (backup path) has a large capacity to make the network more reliable with avoid congestion. On the other hand, we can compute the second shortest path depend on the delay of links in case there is no other path passing through that link. When we said the network must offer convergence in 50ms it means that failure must be detected in less than 50ms, and recovery mechanisms brought up the backup route within 50ms. Then, the service must be restored from end-to-end during this time. My investigation focuses upon how the recovery mechanism includes routing protocol, can recover failure with in less than 50ms. Hence, this process demands additional features to the Dijkstra's algorithm that can compute shortest path faster. In layer3 include the routing protocol, such as (OSPF) with Dijkstra algorithm. In layer2 include the Ethernet, which is concern on Ethernet bandwidth and capacity.

References

[1] A. Shaikh and A. Greenberg, *OSPF Monitoring*: *Architecture, Design and Deployment Experience*, San Francisco, California, March 29-31, 2004, p. 5.

[2] A. Shaikh and A. Greenberg, "Experience in black-box OSPF measurement," in *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW)*, Nov. 2001, pp. 113–125.

[3] A. Shaikh and A. Greenberg, "Optimizing OSPF/IS–IS weights in a changing world," *IEEE*, vol. 20, p. 4, 2002.

[4] Banerjee and D. Sidhu, "Path computation for traffic engineering in MPLS networks," in *Proc. IEEE ICN*, 2001, pp. 302-308.

[5] W. Paper, "Ethernet operations, administration, and maintenance," *Service Management*, 2007, pp. 1-15.

[6] H. Ohta, "Standardization status on carrier class Ethernet OAM," *IEICE Transactions on Communications*, vol. E89-B, no. 3, pp. 644–650, March 2006.

[7] T. Aoyama, "A new generation network: Beyond the internet and NGN," *IEEE Communications Magazine*, vol. 47, issue 5, pp. 82-87, 2009.

[8] J. D'Amboise, "40 Gigabit Ethernet and 100 Gigabit Ethernet: The development of a flexible architecture," *IEEE Communications Magazine*, vol. 47, issue 3, pp. 10-13, Mar. 2009.

[9] H. Ohta, "Standardization status of carrier-class Ethernet," *NTT Technical Review*, vol. 6, no. 2, 2008.

[10] W. Grover, *Mesh Based Survivability Networks*, Prentice Hall.

[11] W. D. Grover, S. M. Iee, D. Stamatelakis, and M. Iee, "Cycle-oriented distributed Reconfiguration," *Cycle*, pp. 537 -543, 1998.

[12] Network Simulator NS2. [Online]. Available: http://www.isi.edu/nsnam/ns

[13] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

Radwan Abujassar is currently assistant professor at the Computer Faculty Department of Arab Open University, Kuwait Branch. Previously, he was an assistant professor at the Faculty of Engineering at the Bursa Orhangazi University in Turkey. Dr. Radwan received his B.Sc. degree from Applied Science University, Amman, Jordan in 2004, and M.Sc. degree from New York Institute of Technology in 2007, both in computer science. His Ph.D. degree in computing and electronic in the field of IP recovery in IGP and MANET networks was received from University of Essex, UK in 2012. His research interests include network and controls, routing protocols, cloud computing and network security.