IP Fast Reroute: Practical NotVia addressing method with Improved ED-NotVia

ABSTRACT

When a failure occurs in a network, it takes routing protocols hundreds of milliseconds to recover. This is intolerable for real-time applications. Thus, many researchers propose the IP Fast Reroute (IPFRR) framework to reduce transmission delay during the failure. As an IPFRR scheme, NotVia provides 100% protection coverage for single-node or single-link failure. However, NotVia introduces nontrivial computing and memory cost. Although Early-Decapsulation NotVia (ED-NotVia) can greatly reduce computing and memory cost compared with NotVia, it still cannot eliminate all unnecessary NotVia addresses.

In this paper, we improve ED-NotVia in two aspects. We first provide the practical NotVia addressing method. With more failure information embodied in the address, a more precise early decapsulation can be achieved, allowing the protection path length to be further shortened. We then redefine the "related-probability" and "related-condition", by which all the unnecessary NotVia addresses are eliminated. We simulate our scheme of Improved ED-NotVia with comprehensive topologies generated by BRITE. The results show that with limited computation overhead, our scheme reduces the memory overhead of NotVia by 53%. Meanwhile, no protection coverage is sacrificed.

1. INTRODUCTION

With the popularity of the Internet, more and more online applications are emerging. Among these applications, the real-time ones, including VoIP, live stream, online gaming, etc., require high-quality transmission service. Although IP routing protocols are designed for robust operation and can recover data transmission when network topology changes, these protocols recover in hundreds of milliseconds or more [?, ?, ?]. But these real-time applications can only tolerate to transmission delay not over tens of milliseconds. Thus, network survivability becomes one of an important topic for today and future network routing.

IPFRR is an IP-based emerging technology that provide protection for data transmission when failure occurs by pre-compute alternative routing paths locally. IPFRR scheme can reduce transmission delay to tens of milliseconds which is acceptable for real-time applications. It can also suppress transient failures, thereby network routing stability can be improved.

NotVia uses special addresses, called NotVia addresses, to encapsulate packet and bypass a failure node or link. In other words, assume that for some source router S, destination router D, next-hop from S to D is F, next-hop from F to D is T (or next-next hop from S to D is T): when the router F or the link from S to F fails, S uses NotVia address to encapsulate packets and forward packets to other neighbor node N instead of F. The packets will reach T not via F, then T will decapsulate the packets and use normal shortest path continue forward packets to D. NotVia address usually be written in form of T_F, which means go to T not via F. NotVia address is an important technique for NotVia mechanism, but there is no practical NotVia addressing method so far [?, ?, ?].

Figure 1 illustrates how NotVia and ED-NotVia work. The shortest path from S to D is S-F-T-R-D. When the link between S and F fails, S cannot send the packet to F, but uses alternative route instead. For NotVia, the protection path from S to D is S-$N2$-R-T-R-D. In this case, transient loop exists. R does not need to forward packet back to T, it can decapsulate packet and use normal route forward packet to D. Thus, ED-NotVia will decapsulate packet when arriving at R and continue forward packet to D, protection path length is shorter than original NotVia. Because NotVia and ED-NotVia treat a failure as single-node failure by default, even if the failure maybe single-link failure. In this example, F itself does not fail, but the link between S and F fails. $N2$ can make earlier decapsulation and uses normal short-

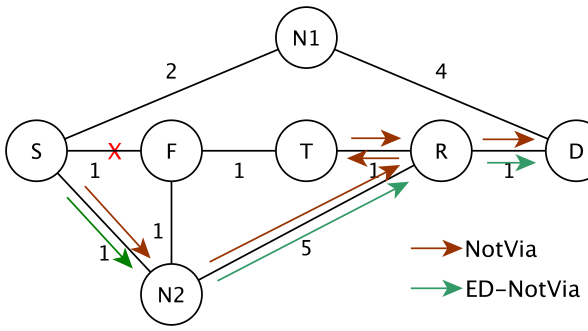


Figure 1: An example of NotVia and ED-NotVia

est path to forward packet via F to D, protection path length will be more shorter. This can be achieved as describe in section 3.

The main purpose of this paper is to improve ED-NotVia in two aspects. First, according to the properties of AS and NotVia addresses, we provide the new practical NotVia addressing method (see below in subsection 3.2). NotVia addresses can carry some useful information, such as failure node F and tunnel-end node T. Thereby, the router R can make early-decapsulation more precise and reduce protection path stretch. Then, we redefine "related-probability" in ED-NotVia algorithm to improve protection coverage with the same parameter p (see section 3.3). And we introduce "related-condition" to our improved algorithm, such that each node will eliminate all unnecessary NotVia address.

We evaluate the performance of our Improved ED-NotVia by simulation based on topologies generated from BRITE. The result show:

- At $p = 1$ (for each node, all other nodes will participate in protection paths computation), Improved ED-NotVia can eliminate unnecessary NotVia address kept by each node up to 53% in average.
- At $p = 1$, 70% single-link failure and 30% single-node failure, protection path stretch will be shorten by 1% to 2% compared to ED-NotVia.
- When threshold p varies ($0 < p < 1$), overall protection coverage slightly better than ED-NotVia.

The remaining part of the paper is organized as follows. Section 2, we further describe NotVia and some improvements, especially ED-NotVia. Section 3, we introduce how we improve ED-NotVia performance as mention above. Section 4, we evaluate the results, compare performance between ED-NotVia and ED-NotVia. Section 5, we conclude our paper in this section. Note that in this paper we only discuss undirected 2-connected topology.

2. BACKGROUND AND RELATED WORK

There are many IPFRR mechanisms, such as Loop-Free Alternate (LFA)[**?**], Tunnel, NotVia[**?**], etc. LFA is one of major research direction of IPFRR mechanisms because it is more lightweight compared to other mechanisms. But LFA fatal disadvantage is that it cannot guarantee protection coverage, the protection coverages are vary depend on topology. In worst case (even number nodes in ring topology), LFA can only provide $\frac{1}{n-1}$ of protection coverage [**?**].

On the other hand, NotVia mechanism can provide 100% protection coverage, but it introduces nontrivial memory and computing cost. NotVia mechanism uses NotVia address to encapsulate packets and bypass a failure. All NotVia addresses in network is two times of number of links, this memory cost is unacceptable for real environment. Because NotVia address T_F means the packets must go to T not via F. So, T_F represents different topology from original one. Any node R needs to exclude F from topology and re-computes SPT for T_F. This introduces nontrivial computing cost to NotVia.

There are some papers discuss how to improve NotVia's performance. In [**?**], authors try to improve NotVia by 3 techniques: NotVia aggregation, prioritization and rNotVia. However, the aggregation approach can reduce only part of unnecessary NotVia address entries, but requires each router be configured with an IP prefix that covers both its interface address and NotVia addresses, introduces new problems to network administrator. In [**?**], authors use concept of redundant trees. Lightweight can significantly decrease the number of NotVia addresses. However, this mechanism requires forwarding two copies of one packet by two different routes, decreasing the efficiency of bandwidth utilization.

ED-NotVia[**?**] is one of NotVia mechanism improvement, it greatly reduce computing and memory cost, but still, ED-NotVia cannot eliminate all unnecessary NotVia addresses. ED-NotVia already introduced "special-condition" and "related-condition" to filter unnecessary NotVia addresses. But ED-NotVia algorithm did not actually use "related-condition" to filter unnecessary NotVia addresses, but use "related-probability" instead. We will discuss more details in subsection 3.3. After using "related-condition" in Improved ED-NotVia, we can eliminate all unnecessary NotVia address.

3. IMPROVED ED-NOTVIA

In this section, we first discuss the properties of NotVia address, traditional NotVia addressing method and our new NotVia addressing method, show how new addressing method help to make decapsulate decision more precise. We then describe how we redefine "related-probability" and how to determine "related-condition". After that, we will provide an Improved ED-NotVia algorithm to save only necessary NotVia addresses and compute protection route for them. Finally, we describe the packets forwarding process with Improved ED-NotVia mechanism.

3.1 NotVia address and traditional addressing method

As mentioned above, NotVia address is the special address used to specify where is next-next hop T of source node (tunnel end) and the failure node F (or the link to F). By NotVia address concept, NotVia address represent different topology from original topology, the only difference is to exclude F from current topology.

Traditional addressing method treats NotVia address like a virtual node and directly assigns an address to this virtual node. The virtual node also needs to advertise its address and routing information over the network, so the normal nodes know the existent and can compute route to the virtual node. Because maximum number of NotVia addresses in topology is two times of number of links, without any aggregation approach, all NotVia addresses needs to be advertised over network, occupy significant bandwidth.

From Figure 1, let some nodes' addresses be: F - 10.0.1.1; T - 10.0.1.2; R - 10.0.1.3. One simple addressing method is directly assigns normal address. To identify NotVia address and normal address, administrator can mark some different NotVia address byte from normal address. For example: T_F - 10.1.1.1; F_T - 10.1.1.2. Or mark some header bits if this is NotVia address. Although, the router can identify if address is normal address or NotVia address, but the router does not know tunnel end T and the failure node F. Because the router cannot get necessary information from NotVia address, so the router cannot make precise early decapsulation. For NotVia address aggregation, network administrator needs to carefully design NotVia address.

3.2 Practical NotVia addressing method

We first analyze the properties of AS:

- Generally, there are hundreds of routers in AS (at most in thousands routers). The number of routers in AS should be less than 65536 (2^{16}).
- Based on IP address assigning properties, first two bytes of address usually are the same (such as 10.0.1.1, 10.0.1.2, etc)
- NotVia addresses are local addresses (loopback), not advertise through outer network.

From our analysis above, our new practical NotVia addressing method is described as below :

1. Discard the same part of address (first two bytes, for example).
2. For NotVia address T_F, use different part of address (last two bytes, for example) to assemble NotVia address in form t1.t2.f1.f2. For example, from Figure 1: T_F - 1.2.1.1; R_T - 1.3.1.2.
3. If NotVia address confuse with normal address, we may mark some bits in the header of NotVia address packet, to identify normal address and NotVia address.

We put T before F because in any node R's SPT T has only one parent node F, when searching FIB, if prefix (first two bytes) are matched, means T_F entry is in FIB, else T_F entry is not in FIB. But F may have many children, to search if NotVia address entry is in FIB or not, all bytes must be matched. Thus, put T before F is better way for FIB searching.

The benefit from our new NotVia addressing method, NotVia address can contain more informations about tunnel end T and the failure node F. The neighbor routers of F can determine if F itself fails or the link S to F fails. Any router R can make early decapsulation more precise if needed.

3.3 Decapsulate condition

Any router R in network will periodically exchange routing information with neighbor routers, neighbor routers know if R still active or not. For example, from Figure 1, when S wants to send packet to D but S cannot currently communicate with F. S encapsulates the packet with T_F address and send packet to *N2*. *N2* receives the packet and checks NotVia address, *N2* knows that tunnel end is T and F still active (only link between S and F fails). In this case, the failure type is link failure not node failure, *N2* can make a decision to decapsulate the packet or not.

After F neighbor router R receives packet with NotVia address and identifies the failure type is link failure, when R can make early decapsulation? Actually, two possibilities exist after R decapsulate the packet:

- F is not in *route(R, D)* (shortest path from R to D). R can make early decapsulation and use normal path to forward the packet. The packet will not be sent back to S and back to R. Because *route(S, D)* pass through F, contradicts with the premise. In this case, R can make early decapsulation and forward packet.
- F is in *route(R, D)*. After decapsulation, the packet maybe forward back to S or maybe not. To prevent the packet to be sent back to S and back to R itself, the relation among S, F and R must satisfy below condition:

$$d(R,\ F) < d(R,\ S) + d(S,\ F) \qquad (1)$$

Before decapsulating the packet, R does not know which is exact destination D. According to above analysis, we only need to worry about F in *route(R, D)* case, decapsulation will be problem only if S, F and R do not satisfy equation (1). To be easier for implementation, we may use "*route(R, F)*'s next hop is F itself" instead of equation (1).

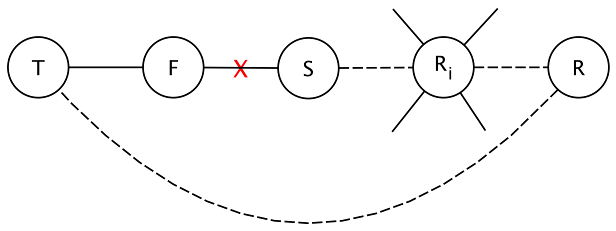


Figure 2: ED-NotVia's "related-probability"

Thus, sum up with conditions introduced in ED-NotVia, Decapsulate conditions are:

1. T_F does not exist in NotVia FIB. (from ED-NotVia)
2. R is F neighbor router and equation (1) is satisfied. (new condition)

We will describe packet forwarding process in subsection 3.6.

3.4 "related-probability" and "related-condition"

ED-NotVia proposed "special-condition" and "related-condition" to reduce unnecessary NotVia addresses:

- special-condition: if the next hop of *route(R, T_F)* is the same with the next hop of *route(R, T)*, then T_F is ordinary to R; else, T_F is special to R.
- related-condition: if R is not in any protection path from any S to T_F, then T_F is unrelated to R; else, T_F is related to R.

ED-NotVia algorithm used "special-condition" to filter unnecessary NotVia address, it did not use "related-condition", but used "related-probability" instead.

"Related-probability" uses the intuitive concept that the probability of farther nodes' NotVia address sent to R is quite small, and vice versa. As shown in Figure 2, after S detect a failure and encapsulate packet, assume there are k nodes (R_i) between S and R, and R_i's degree is d_i. So the probability that T_F will be sent to R is $1/(d_1 * d_2 * ... * d_k)$ (the probability at R_i is $1/d_i$).

Although this intuitive concept may not be correct in all cases (ring topology, for example), evaluation result from ED-NotVia and our Improved ED-NotVia show that this concept works very well in practical. Hence we decide to continue using this "related-probability" concept. We redefine "related-probability", not only use node's degree in probability computation but also consider the weight of link in probability computation. Because in real environment, the cost of links in topology is not uniform, the links on shortest path should be smaller compared to others. Thus, our "related-probability" at R_i define as follow:

$$q_i = \frac{\frac{1}{w_i}}{\frac{1}{w_1} + \frac{1}{w_2} + ... + \frac{1}{w_k}} \tag{2}$$

We consider using "related-condition" in Improved ED-NotVia algorithm. Because in undirected graph, the shortest paths from any node A to a node B is the same SPT rooted at B. To check if R is in any protection path *route(S, T_F)*: exclude F from topology and compute SPT rooted at T. If R has children, it means R is in the shortest path from its children to T. In the other words, R is related to T_F. We call the SPT rooted at T as "reverse-SPT" or "rSPT".

3.5 Algorithm

Now, our Improved ED-NotVia algorithm to find all necessary NotVia addresses and compute protection paths described as follow:

Algorithm 1 Improved ED-NotVia

```
Require: $R$, SPT rooted at $R$ and threshold $p$
Ensure: all necessary NotVia addresses and protection paths
 1: if $0 < p < 1$ then
 2:     $threshold \leftarrow 1 - p$
 3: else
 4:     $threshold \leftarrow 0$
 5: end if
 6: $root \leftarrow SPT.root, root.q \leftarrow 1$
 7: for all $ch \in root.children$ do
 8:     $ch.q \leftarrow q_i$ // from equation (2)
 9:     $\Phi.enqueue(ch)$
10: end for
11: while $\Phi.empty() == false$ do
12:     $F \leftarrow \Phi.dequeue()$
13:     $iSPT \leftarrow iSPF(SPT, F)[?]$ // SPT without $F$
14:     for all $T \in F.children$ do
15:         if $F.q < threshold$ then
16:             continue // not related(probability)
17:         end if
18:         $T.q \leftarrow q_i * F.q$
19:         $\Phi.enqueue(T)$
20:         $nexthopT_F \leftarrow getNextHop(iSPT, T)$
21:         if $T.nexthop == nexthopT_F$ then
22:             continue // not special
23:         end if
24:         $rSPT \leftarrow SPF(T, F)$ // compute SPT rooted at $T$ without $F$
25:         if $R \in rSPT, R.children.empty() == true$ then
26:             continue // not related(conditon)
27:         end if
28:         $R.NotViaFIB.add(T_F)$
29:     end for
30: end while
```

Figure 3 shows SPT rooted at R. The algorithm is to traverse through SPT by breadth-first order, starting from children of R. Start from A, we assume A

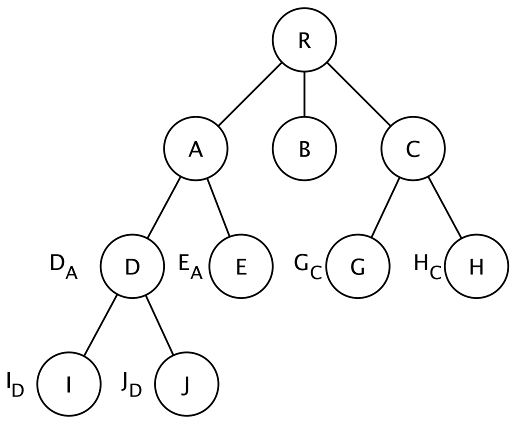


Figure 3: SPT root at *R* with NotVia addresses

is the failure node, then compute iSPT without A in topology. If D_A satisfy "related-probability", enqueue D's children; else go to E. If D_A satisfy both "special-condition" and "related-condition" then add D_A to R NotVia FIB; else discard this NotVia address.

3.6 Packet forwarding process

The packet forwarding process of Improved ED-NotVia is described as follow:

1. If the packet destination address is NotVia address, then go to step 4.
2. If destination address is the same as current node address. The packet is already reached destination. End process.
3. From normal FIB find next hop. If next hop is reachable, then forward the packet and go to step 7; else, encapsulate the packet with NotVia address.
4. (NotVia address case) If satisfy one of conditions below, then make decapsulation :
    - Already reach tunnel end.
    - This T_F does not exist in NotVia FIB.
    - This node is F's neighbor, F itself didn't fail and satisfy equation 1.
5. From NotVia FIB find next hop, forward the packet and go to step 7.
6. Decapsulate the packet. From normal FIB find next hop. If next hop is reachable, then forward the packet; else, drop the packet and end process.
7. If successfully forward packet, then decrease TTL by 1. Repeat forwarding process.

4. EVALUATION

In this section, we show how our new practical NotVia addressing method benefits to NotVia mechanisms and compare our Improved ED-NotVia performance with original ED-NotVia.

4.1 Methodology

We use C++ to implement the simulator that computes FIBs, generates failures and simulates packet forwarding process. First step, we compute normal FIBs, ED-NotVia FIBs and Improved ED-NotVia FIBs from topologies generate by BRITE. Second step, we simulate a failure in network by randomly pick 400 sets of $\{S, D, F\}$ for each topology. According to [?], in real world more than 70% of failures are single-link failures. Thus, our simulation will use proportion of failures : 70% are single-link failures and 30% are single-node failures. Since we only discuss 2-connected topology, we will discard the set of $\{S, D, F\}$ if exclusion of F affected topology connectivity. Last step, we simulate the packet forwarding process by using loop lookup next hop from FIBs hop-by-hop, to ensure that our simulation would be as real as possible.

We compare the performance of ED-NotVia and Improve ED-NotVia in 3 aspects: threshold p and protection coverage relation, unnecessary NotVia addresses eliminated by "related-condition" and protection path stretch.

4.2 p and protection coverage

ED-NotVia and Improved ED-NotVia use threshold p to determine when to discard low "related-probability" NotVia addresses. In the other words, while traversing R's SPT to find NotVia addresses and compute protection path process, if some A's q value is less than threshold p, A and its children would not be enqueued to checking queue and be discarded.

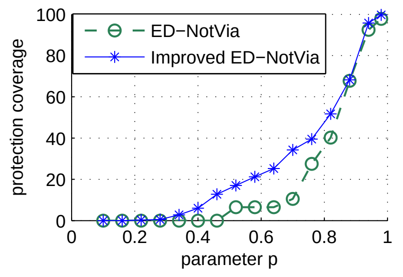
Since low "related-probability" NotVia addresses do not mean these NotVia addresses are unnecessary, the protection coverage depends on threshold p. If p is too small, there should be many wrong regarded-as-unnecessary NotVia addresses. If p is 1, this means all necessary NotVia addresses will be kept. In real situation, the network administrator can do a trade-off by sacrificing a little protection coverage to gain more memory space, or vice-versa.

In this test, we set p value vary from 0 to 1 (excluded) in different topologies and observe protection coverage percentage.

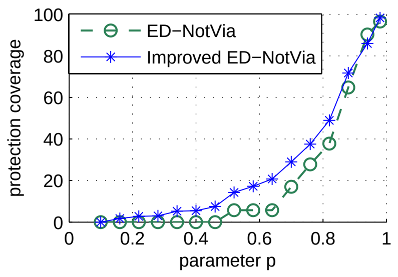
As shown in Figure 4, Improved ED-NotVia's overall protection coverage is slightly better than ED-NotVia. Because, protection coverage depends on the number of necessary NotVia addresses kept by R, so our "related-probability" perform better in pruning regarded-as-unnecessary NotVia addresses with only little improvement.

4.3 Unnecessary NotVia addresses

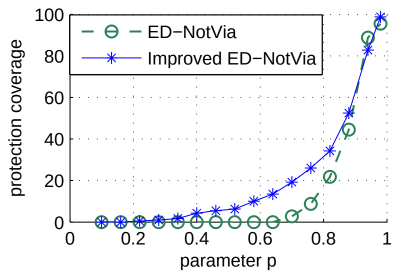
In this aspect, we use unnecessary NotVia addresses to measure memory cost efficiency. Unnecessary NotVia addresses ratio is the number of unnecessary NotVia addresses divide by the number of all NotVia addresses



(a) n=50, m=2

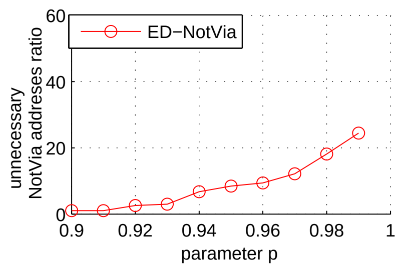


(b) n=80, m=2

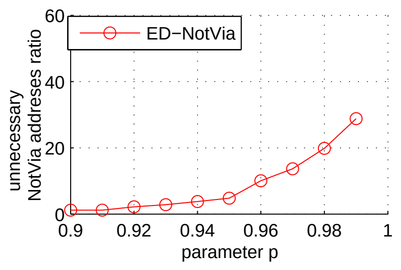


(c) n=120, m=3

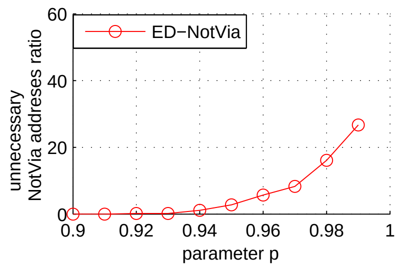
Figure 4: *p* and protection coverage



(a) n=50, m=2



(b) n=80, m=2



(c) n=120, m=3

Figure 5: *p* and average unnecessary NotVia addresses ratio

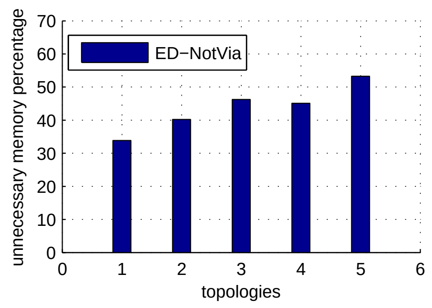
kept by current node. Thus, unnecessary NotVia addresses ratio directly represents current node memory utilization.

Figure 5 shows threshold p starting from 0.9 to 1 and unnecessary NotVia addresses ratio relation. While p value is small, only NotVia addresses"near" root R kept, and most of them are necessary NotVia addresses. When p value becomes bigger, the number of unnecessary NotVia addresses becomes bigger too.

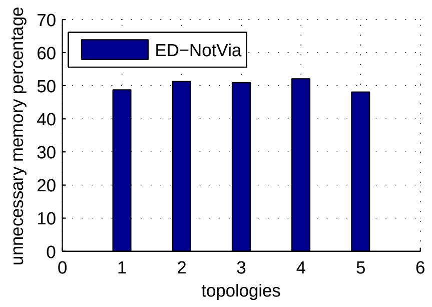
Figure 6 show unnecessary NotVia addresses ratio of ED-NotVia without pruning some NotVia addresses by "related-probability" (p=1). We first test unnecessary NotVia addresses ratio of different topologies with different number of nodes and links. When topology become larger (more nodes and links), unnecessary NotVia addresses ratio also increase. We then study NotVia addresses ratio of different topologies with the same number of nodes but different number of links. From the result, there is no significant different between these topologies. Thus, the number of nodes should affect unnecessary NotVia addresses ratio more than the number of links. Our Improved ED-NotVia can save memory up to 53% compared with original ED-NotVia mechanism.

4.4 Protection path stretch

By introducing our new NotVia addressing method, the neighbor routers of F can identify failure type. In

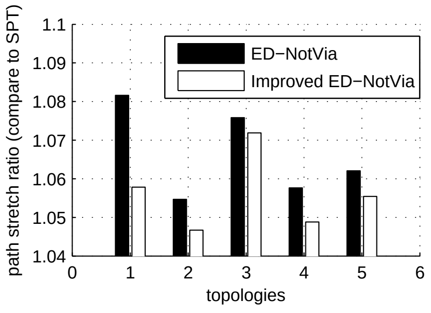


(a) 1.n=50 , m=2; 2.n=80 , m=2; 3.n=120 , m=3; 4.n=150 , m=4; 5.n=280 , m=4

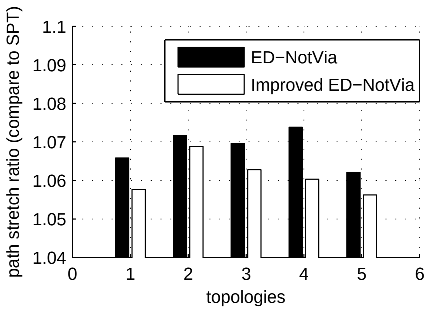


(b) 1.n=250 , m=2; 2.n=250, m=3; 3.n=250, m=4; 4.n=250 , m=5; 5.n=250 , m=6

Figure 6: average unnecessary NotVia addresses ratio on different topologies (p = 1)



(a) 1.n=50 , m=2; 2.n=80 , m=2; 3.n=120 , m=3; 4.n=150 , m=4; 5.n=280 , m=4



(b) 1.n=250 , m=2; 2.n=250, m=3; 3.n=250, m=4; 4.n=250 , m=5; 5.n=250 , m=6

Figure 7: average protection path stretch on different topologies ($p = 1$)

single-link failure cases (more than 70% of all failures) [?], we can make decapsulation more precise and get benefit from our new NotVia addressing method.

Protection path stretch is the ratio between protection path length and the exact shortest path length after exclude F from topology. From Figure 7, although original ED-NotVia mechanism performs well in reducing protection path stretch to below 10%, but our Improved ED-NotVia can futher shorten protection path stretch around 1% to 2% of protection path stretch without sacrificing any other performance. From the result, there is no direct relation between topology's size and protection path stretch reduced.

5. CONCLUSION AND FUTURE WORK

In this paper, we have presented the new practical NotVia addressing, which can fulfill current NotVia mechanisms and help Improved ED-NotVia mechanism to identify a failure type and reduce protection path stretch. We also proposed Improved ED-NotVia, which is the further improvement over NotVia mechanism. Improved ED-NotVia is better than original one in three aspects: first, overall protection coverage is slightly better than ED-NotVia's; second, with some time cost for rSPT computation, unnecessary NotVia addresses can be eliminated up to 53%; third, in single-link failure case, we can get benefit from new addressing method and further reduce protection path stretch.

Improved ED-NotVia is an easy-to-implement and the practical NotVia approach to enhance network survivability and QoS. However, Improved ED-NotVia needs more time cost to reduce memory cost of NotVia addresses. If possible, we will optimize rSPT algorithm to decrease computation cost. As the next step, we will discuss and sum up with other technique, to find better IPFRR solution.

6. ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers and ED-NotVia's authors for their valuable comments.

7. REFERENCES

[1] G. Enyedi, G. Rétvári, P. Szilágyi, and A. Császár. IP Fast ReRoute: Lightweight Not-Via. In *Proceedings of the 8th International IFIP-TC 6 Networking Conference*, NETWORKING '09, pages 157–168, Berlin, Heidelberg, 2009. Springer-Verlag.

[2] G. Iannaccone, C. nee Chuah, S. Bhattacharyya, and C. Diot. Feasibility of IP restoration in a tier-1 backbone. *IEEE Network*, 18:13–19, 2004.

[3] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. *IEEE/ACM Trans. Netw.*, 9(3):293–306, 2001.

[4] A. Li, P. Francois, and X. Yang. On Improving the Efficiency and Manageability of NotVia. In *Proceedings of the 2007 ACM CoNEXT conference*, CoNEXT '07, pages 26:1–26:12, New York, NY, USA, 2007. ACM.

[5] Q. Li, M. Xu, Q. Li, D. Wang, and Y. Cui. IP Fast Reroute: NotVia with Early Decapsulation. In *GLOBECOM'10*, pages 1–6, 2010.

[6] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot. Characterization of failures in an operational IP backbone network. *IEEE/ACM Trans. Netw.*, 16:749–762, August 2008.

[7] P. Narváez, K.-Y. Siu, and H.-Y. Tzeng. New dynamic SPT algorithm based on a ball-and-string model. *IEEE/ACM Trans. Netw.*, 9:706–718, December 2001.

[8] G. Rétvári, J. Tapolcai, G. Enyedi, and A. Császár. IP Fast ReRoute: Loop Free Alternates Revisited. In *Proc. IEEE INFOCOM*, Shanghai, P.R. China, 4 2011.

[9] M. Shand, S. Bryant, and S. Previdi. IP Fast Reroute Using Not-via Addresses. IETF Draft, March 2010.

[10] A. Zinin and A. Atlas. Basic Specification for IP Fast Reroute: Loop-Free Alternates. IETF Draft, September 2008.