Network Failure Analysis for CENIC

Yunqi Zhang
UC San Diego
yqzhang@ucsd.edu

Wangfan Fu
UC San Diego
wafu@ucsd.edu

Dexin Qi
UC San Diego
deqi@ucsd.edu

***Abstract*—The network services today are having a significant increase of the requirement on the availability, while at the same time the growing size and complexity of the networks we are building makes it even more challenging to provide such property. In order to provide better network avilability we will need to reduce the failures, that we believe a better understanding of the failure itself should be considerably helpful. However, the high expense and performance overhead of the traditional approaches eliminates the feasibility to perform such study across the production network.**

We propose a much cheaper approach to deliver such failure analysis by carefully extracting useful information from syslog, which is commonly provided by most production networks today. And we present a methodology to analyze and categorize the unexpected network behaviors including routing loops and non-existent links. In addition, we evaluate the performance of the routing protocols on calculating the shortest paths and adapting to the failures. To validate our mechanism, we carry out a series of analysis in CENIC, which is fairly large and general.

I. Introduction

As more and more enterprises are heavily relying on the networks, there is clearly an increase in importance to provide better availability, or even uninterrupted service more aggressively. However, it brings us a much more challenging work to provide such promise as we are building much more complex and larger networks comparing to years ago. Although tons of research projects have been trying to work it out, it is still not very well solved. As well as many previous work shown [6], [7], [9], we believe the better analysis of the network failures should be considerably helpful for us to understand how the failures occur, how they affect the network behaviors and how to eliminate them in order to provide better availability.

To deliver such network failure analysis in practice, one will need large amount of information about the causes, time lasted, influence to the network and many other specifics that are clearly not intuitively provided by the network protocols in use today. Thus, the traditional approaches [4], [10] for doing this kind of analysis is to build some extra software or hardware to probe the network condition actively, which would incur large amount of expense and perhaps performance overhead. Consequently, most of such failure analysis are performed in the research community. To our knowledge, California Fault Lines [12] is the first research project that trying to extract these information from commonly used production networks today.

Other than reconstructing historical network failure events, we are able to perform further characterization and analysis about network failures with very similar ”cheap and dirty” data which we can easily obtain from today’s networks. To better understand the network behavior, we describe and validate a methodology to map the routing changes to corresponding network failures to understand how failures affect the network behaviors. In addtion, characterization and discussion are presented about the unexpected network behaviors including routing loops and non-existent links. Furthermore, we analyze the performance of the routing protocols on how well they are doing in calculating the shortest path and how fast they could adapt to the link failures.

Specifically, we set up six source machines respectively at UC Berkeley, UC San Diego, UC Los Angeles, UC Santa Barbara, UC Davis and UC Santa Cruz, all in CENIC (the Corporation for Education Network Initiatives in California) which is the autonomous system we are trying to analyze. Each of the six machines are supposed to send a series of traceroute requests to the end hosts of the failing link when it is detected with the help of syslog messages automatically sent by CENIC. Firstly, we map the traceroute data back to corresponding network failures for further study, and validate how well the failure detection and traceroute are working based on the mapping results. Secondly, we detect the routing loops and non-existent links in the traceroute data, and carry out some statistics and characterization on these unexpected network behaviors as well as the network failures. At last, with the help of link weights of CENIC, we compare the optimal shortest route and the actual one described by traceroute, and evaluate how well the network protocols are working and analyze how the failures affect the network behaviors.

The rest of the paper is organized as follows. We begin with the discussion of related work in Section II. Then the data we use in this work is introduced in Section III. Section IV presents the our methodology for the characterization and analysis, and they are validated in Section V. We present our analysis in Section VI and conclude in Section VII.

II. Related work

Network failure analysis has long been an interesting topic since the born of the Internet [1]. Researchers have done a lot work in charaterizing all kinds of network failures [3], [5], [8], [11], [12]. These research projects all did thorough studies on the network failures from differenct perspectives.

In [8] Athina and his colleagues analyzed failures in a IP backbone operated by Sprint.Based on the IS-IS data they got from the backbone network, they categorized these failure data by causes, and did some statisitic work on these data. However,

they did not dig too much into the acutal characteristics of different failures.

While in [5], [11], researchers carried detailed analysis under large enterpise data centers. In [11], Shaikh and his colleagues focused on OSPF behavior. They tracked 205 routers for a month, described how LSAs works, and gave convincing corresponding explainations. Although they are intended to gave a clear illustration of how LSAs behaves, they offered a valuable insight of network failures at the same time. On the other hand, in [5], Phillipa and his colleagues aimed to find out how failures behaves inside a larege enterpise data center. Since they got access to the data center directly, they were able to classify all kinds of hardware and software failures. They gave a lot of statistics about how failures happened, and how redundancy performed in dealing with these failures.

The work mentioned above all involved hardware or software infrastructures to probe the network condition actively. In [12], Daniel together with other researchers did a detailed analysis about failures over the CENIC network, with only the syslog history, mailing list and configuration setting of the involved routers. They successfully reconstructed the history of the network and found many interesting behaviors.

In contrast, we continued to work on the data extracted from commonly available sources of modern network, and presented a analysis than is generally applicable on most production networks today. We believe our study is the first effort that focuses on analyzing and characterizing the network behaviors with such dirty but cheap data.

III. Data sources

We delivered this piece of work in CENIC, from which we got several different sets of data. In order to set the context for our further analysis and characterization, we described the particular data sources we used as follows. We extracted these data to make it as neat as we could since the raw data was much more messy then described below.

A. The CENIC Network

The Corporation for Education Network Initiatives in California (CENIC), was a state-wide network providing Internet access to California's education and research communities. [2] It had a combined enrollment of more than six million members including the University of California system, California State University system, community colleges, and K-12 school districts.

B. ISIS Failure

This failure data was detected by the underlying routing protocol based on the adjacency status. The protocols running across the ISs required the routers to send and receive *hello* messages. By default, each router were supposed to send a *hello* message to its neighbors every 10 seconds. And if one router hadn't heard from one of its neighbors for 30 seconds, it was declared as disconnected. Thus a IS-IS failure was reported as the link between the router and its neighbor. After the router heard from the failed neighbor again, the failure was marked as resolved.

For each of these failures, it followed the format shown below. The fields $failure_{start}$ and $failure_{end}$ were formatted in UNIX timestamp.

$\{router_1, port_1, router_2, port_2, failure_{start}, failure_{end}\}$

C. IP-Router Mapping

As different IP addresses were assigned to different routers at different time, there was a table describing the mapping between IP address and router at some point of time. And each record contained a field showing the IP address assignment was valid until a certain time. Consequently, the assignment became valid right after the last assignment of the same IP address became invalid.

For each record, it followed the format shown below. The field $time_{valid\_till}$ was formatted in UNIX timestamp.

$\{IPaddress, router, port, time_{valid\_till}\}$

D. Link Map

The link map described the available links in CENIC and their valid time period. Each of the record contained the two routers and two ports the link connected, and the valid time period. If there wasn't a record for two particular routers, it meant that they were not connected.

For each link, it followed the format shown below. The fields $link_{start}$ and $link_{end}$ were formatted in UNIX timestamp where the links that are still valid right now were marked huge numbers in $link_{end}$.

$\{router_1, port_1, router_2, port_2, link_{start}, link_{end}\}$

E. Link Weights

The link weights were presented with the two routers it connected and one integer weight value. The weights were typically assigned to the links by configuration files, and the values were not consecutive. The smaller the weight is, the more likely the routers would select it as a shorter path.

For each record, it followed the format shown below.

$\{router_1, router_1, weight\}$

F. Traceroute

We had 6 machines set up at different locations in CENIC, and they were supposed to send traceroute request to the ends of the failure link when it was detected. The reason why we had 6 of them was because the more source machines we had, the more likely that we could get aware of the routing changes incurred by the link failures, because different sources would send the requests from different directions of the network and some of them might not route through the failure link so they wouldn't be affected by it actually. Those machines were located respectively in UC Berkeley, UC San Diego, UC Los Angeles, UC Santa Barbara, UC Davis and UC Santa Cruz.

The traceroute data were simply the original output of the traceroute application. However, there were two issues we needed to pay more attention in order to get more accurate characterizations.

1) Hops with local area network The source machines we set up were within particular universities. Typically, the first few hops of the traceroute were within the local area network, which we were not actually interested in because they were not involved in CENIC actually. Consequently, we needed to verify whether the hop was within LAN or in CENIC by looking up if there was an IP address assigned to it.
2) Unrecognizable hops Not all the routers in CENIC support the UDP traceroute application we used to get the actual routing information, so some of them would just reponse as ** when they were traced. We were not able to get rid of it, thus we had to be careful with it in the following analysis.

IV. Methodology

A. Reconstruction of the Network History

Before doing any other analysis or characterizations, we needed to reconstruct the network history and topology from the data sets we had. First of all, we parsed the raw data of ISIS failures, mapping of router and IP address and available links within CENIC. Most of these were timely, that for instance one IP address could be assigned to many other routers or end hosts after it expired for one router. So we stored these data with timestamp at the granularity of one second, which was exactly the granularity provided by the raw data, and kept track of them timely.

Then we parsed the traceroute data, which was a little bit more complex since there were many trivial issues we needed to take care.

1) For most of the traceroutes, the first few hops were not actually the routers or end hosts in CENIC, instead they were basically the routers within the local area network which we were not actually interested in as we mentioned ealier.
2) In addition, there were some routers in CENIC that didn't support UDP traceroute, which meant the traceroute could give us nothing more than ** showing that the router was not recognizable.
3) And there was also an issue about the destination of traceroute, it turned out that the traceroute stopped when it reached the same router of the destination regardless of the port indeed. So we would need to check whethre the last hop of traceroute and the destination are on the same router rather than exact same port and IP address.

B. Mapping between ISIS Failures and Traceroutes

We would like to map the ISIS failures to corresponding traceroutes and vice versa not only to validate how the failure detection and traceroute were working, but also for further characterization and analysis. In order to do so, we had two rules to establish the correlation between them. Based on the mapping information, we were able to compare the traceroutes data and looking for the changes and which failure caused them.

1) Time Consistency The occurrence of the failures and their corresponding traceroutes should be fairly close in time. In particular, because of the possible 30-second delay of our failure detection mechanism, we had to widen the mapping window on time consistency.
2) Route Consistency This one was a little bit trickier because the destination of the traceroute and the failure link could not be the same port and IP address, so we had to check whether they were supposed to be on the same router.

C. Non-existent Link Detection

We then extracted the nonexistent links from the traceroute data, as a part of the unexpected routing behavior. We traversed through all the traceroute data, assuming there should be a link between any adjacent hops in the records. This is verifried by looking up in the linkmap to see if there is a link between these two routers at that time. Although it seemed to be fairly straight forward, we have to apply some constraints to the process in order to avoid useless data including time and route consistency.

1) Look up by router We could say there was a link between the two hops if there was a link between the two routers on the hop, without considering the port. Since the weights used to calculate the route in the network is given by routers, ignoring ports would eliminate useless data. In addition, traceroute checked router, instead of router with port, to see if it had reached the destination too.
2) No respones During this process, we ignored those hops that failed to respond to traceroute, which appeared in the record in the form of **, as well as those routers out of CENIC network.

D. Routing Loop Detection

We also extracted routing loops from the traceroute data, as the other part of the unexpected routing behavior. We parsed all the traceroute records and detected loops following two different rules as described below.

1) Verify by IP We could verify whether a traceroute record contained loop by checking if there existed an IP address appearing more than once.
2) Verify by router We could verify whether a traceroute record contained loop by checking if there existed a router appearing more than once. This approach should at least return all records that previous approach returned.

E. Routing Performance Characterization

Basically, we were focusing the performance on calculating the shortest routing path by saying routing performance. Given the link weights of each links in the CENIC, we were able to calculate the sum of the weights for each of the traceroute record while regardless of the first few hops within their local area network, as well as the weight of the global optimal one.

However, there were some issues we needed to pay attention for both of these calculations.

1) Actual route There might be some of the middle hops were not recognizable because lack of the support of UDP traceroute. For these records, the best effort we could do is to calculate the weight of the sub-route. However, this didn't make much sense in practice because there was a great probability that the actual route and optimal one had totally different sub-routes, so that they were not comparable at all.
2) Optimal route Here we just simply implemented the Dijkstra's Algorithm to calculate the optimal shortest path.

In addition, we had to get rid of the first few hops within local area networks for both optimal route and actual one since we didn't have the link weights of them.

Then with 70831 traceroute records, we measured the time each record lasted as an indicator of congestion condition in the network. We then plotted a graph to show the trend between traceroute lasting time and the number of failures with respect to time. We had normalized the failure data, and cutted some extremely high spike. We could see that they share a similar trend there in Figure 1, which meant it was likely that the failures of links did affect the congestion condition of the network.

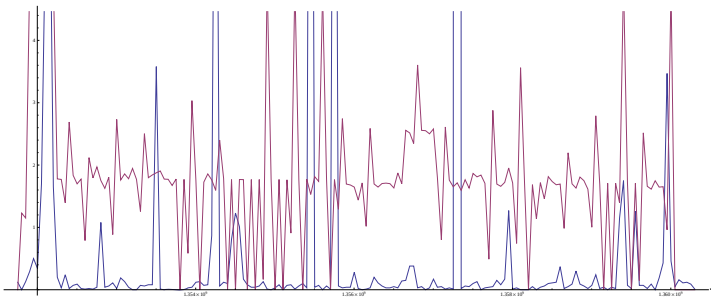

Fig. 1. Comparison between traceroute performance and the number of failures in the network. The red line was the normalized lasting time of the traceroute request, which meant how much time it took to finish one request. And the blue line was the normalized number of failures happening in the network at same time. What we could tell from this figure is that they were very likely to follow the similar trends.

V. Validation

A. Network History

Assuming the failure detection of the certain network could give us a complete history of it, we had this history of the CENIC from 10/20/2012 to 02/09/2013. There were 40074 ISIS failures during this period in total.

Since we would like to do a comparison among the route before, during and after the failure to better understand how failure affected the network behaviors, it was useful to find out how long each failure lasted. Thus, the CDF of failure recovery time was shown in Figure 3. Surprisingly, there were 77.87% of the failures that were recovered in 0 seconds, which means they didn't actually affect the network in the granularity

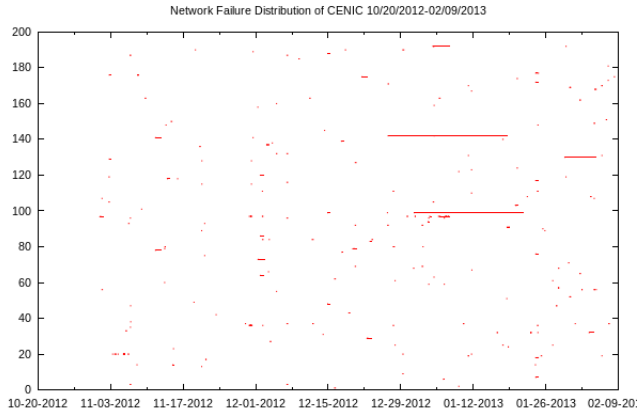

Fig. 2. Network failures distribution of CENIC from 10/20/2012 to 02/09/2013, where Y-axis is the failure link. Most of the failures lasted for a fairly short time period and there were 192 links had failed at least once during this time period.

of second that we used. And more than 95% failures were recovered within 6 seconds, which could potentially make the corresponding traceroutes much less interesting.

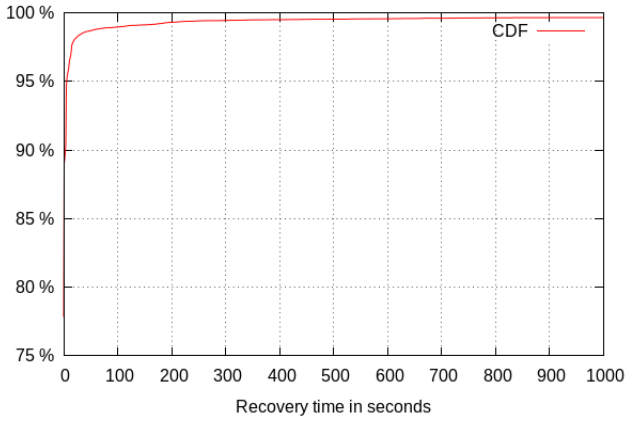

Fig. 3. Network failure recovery time CDF of CENIC where more than 95% failures recovered within 6 seconds. The short recovery time was a good indicator for better network availability, while on the other hand reduced the number of interesting data we could analyze since a large portion of the failures had already recovered before we were able to send out the traceroute request.

Because of the relatively short recovery time, it is very likely that the failure had already recovered before we sent very few traceroutes. Thus it could reduce the amount of data we were interested in because most of the traceroute would not even aware of the route changes due to the large granularity. However, even with small percentages, we didn't get very tiny numbers in the following analysis because of the fairly large number of failures we had in total.

B. Network History Reconstruction

To validate the network history we reconstructed from the traceroute data, we tried to map the probed failures to corresponding traceroutes. Based on the link and router map

of CENIC we had successfully mapped 40073 ISIS failures to corresponding traceroute data out of the total 40074. The only one that had not been mapped was because we didn't have the IP address of the link which meant the router was probably not within CENIC. Thus, we believed it was save to say our failure detection worked pretty well from this point.

For these 40073 detected failures, we had pinged at least one end of the failure link. And for 39505 out of 40073 (98.58%) failures, we had pinged both ends at least once. Consequently, we believed our data is very representative for our analysis.

C. Statistices of Unexpected Network Behaviors (Loops & Non-existent Links)

Trace Route data(70831)			
Loops(366)		Nonexistent Links(80)	
Reached	Not Reached	Reached	Not Reached
11	355	69	11

Fig. 4. Statistics of Unexpected Network Behavior

From Figure 4, we could find that loops have very different charasteristics comparing to non-existent links. Loops happened much more often than non-existent links, and were much more likely to cause a failure in traceroute records. On the other hand, most of the traceroute with non-existent links could still get to the destination at last. This pointed out that, routing loop was often a great threat to a network, and should always be avoided if possible.

D. Measurement Bias

First of all, the granularity of the network failures and the traceroute data could potentially affect our analysis of the network behaviors. As discussed ealier, the ISIS failures were probed at the granularity of one second, while many of them lasted less than couple seconds actually. This reduced the number of interesting traceroutes since they could not even reflect the effects of such short failures. We believed that a much finer granularity of the failure detection would be very helpful for further analysis.

Additionally, our failure detection mechanism was based on adjacency status reported by the underlying routing protocol. For example, to ensure connectivity, the IS-IS protocol required routers to send and receive *hello* messages. By default, a router sent a *hello* message once every ten seconds and declared a link disconnected if no *hello* message was received for thirty seconds. Consequently, the failure detection could have up to 30 seconds of delay, which could reduce the accuracy of the mapping between the failure and corresponding traceroutes. In addition, this would probably affect our analysis about how the failures affected the network behaviors.

VI. Data Analysis

A. Loop Characterization

In the 70831 traceroute records, we detected 366 records by IP approach and 395 records by router approach that contain loops in total.

For IP approach, we categorized these 366 records by the length of loop and found that more than 95% (350 out of 366) of loops were jumping between only 2 routers. The longest loop had a length of 8. As of the long loop records, at least 11 out of 16 would also fall into a two-router trap finally. We also found that there were only 11 rcordes that arrived at the destination eventually. All of the 11 records shared loop length of 2. And 9 out of 11 records only experienced 1 round of loop. Another interesting phenomenon was that 344 of 366 records were trapped into a loop that consisted of only 2 adjacent IP addresses.

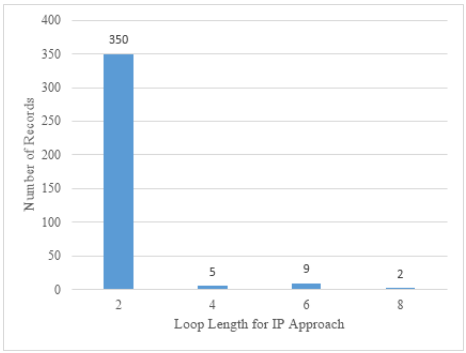

Fig. 5. Loop length statistics. Most of the routing loops were fairly short that 350 out of 366 (about 95%) had a length of 2. And the longest loop had a length of 8 partly because the traceroute we sent had a constraint of maximal number of hops as 15.

For Router approach, we categorized these 395 records by the length of loop and found that more than 90% (359 out of 395) of loops are jumping between only 2 routers. As of the long loop records, at least 24 of 36 would also fall into a two-router trap eventually. We also found that there were 29 records that finally arrived at the destination. 26 of the 29 records shared loop length of 2 while 2 records had length of 4, and 6 for the last one. 26 of 29 records only experienced 1 round of loop.

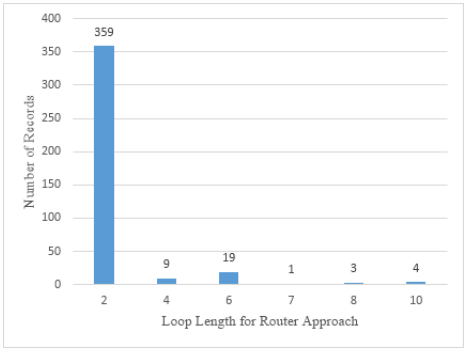

Fig. 6. Loop Length Statistics

We then tried to figure out a map from loop records to IS-IS failures to reveal what caused such unexpected routing behavior. We categorized the loop records by the potential pattern of failures' impact. By applying both time consistency and router consistency, we obtained the first category, Simple

Link failure. However this pattern could only explain a small portion of the loop records. Introducing a failure propagation assumption by loosing the time constraints granted us the second category, Complex Failure Propagation. Potential causes for the rest part of records that couldn't be explained by the two reasons above might be done in futurte work. For the following part, we only consider the 366 records we obtained by IP approach.

Simple Link Failure

Suppose a trace-route arrived at destination without any unexpected failures. A simple link failure may occur in any links on the normal path. Then the router would have to forward packets to other routers which actually could not help reach the destination and finally result in a loop.

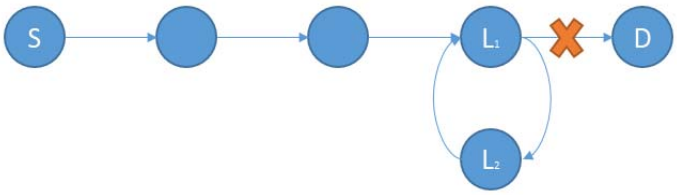

Fig. 7. Simple link failures, where the newly introduced hop did really help to forward the packet to the destination, so the last two hops fell into a small loop.

Complex Failure Propagation

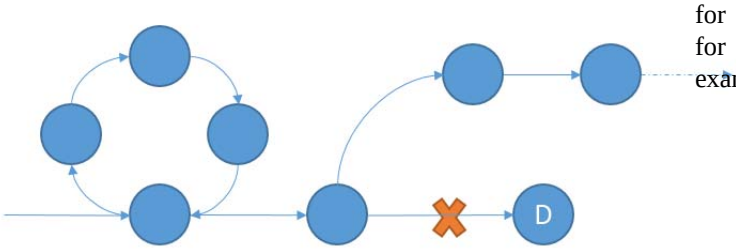

Fig. 8. Complex failure propagation, where none of the alternative routes could work out to forward the packet successfully.

While considering the direct influence of failure in the above part, it was also possible that some failures might leave impact on the network even after they were revocered. Suppose a traceroute arrived at destination without any unexpected failures and then a failure occurred at one link on the path. Then the routers ahead the failure node might detect that forwarding packets to the failure node was useless. Eventually these routers would try some other routes. Unfortunately sometimes none of these routes worked at all, then a loop would happen again.

B. Non-Existent Links Charaterization

Among the 70831 trace-route records, there were 80 records had non-existent links in them. We classified these records into 5 categories, with respect to the complexity of their cause.

1) normal route shift
2) complex route shift
3) shortcut
4) trapped in loop
5) cannot verified

They are distributed as Figure: 9 shown.

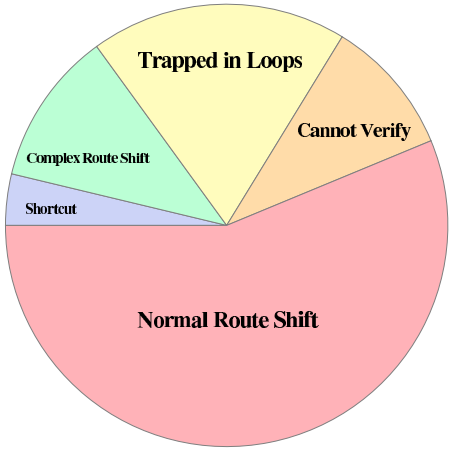

Fig. 9. Distribution of the non-existent links, where more than half of them were normal route shifts.

Normal Route Shift

Suppose we had a traceroute with 5 as the max number of hops. In most situation, a certain route wouldn't change during the process of traceroute because the traceroute lasted for relatively short time period. However, it was possible for routers to choose another route during this process. An example is shown in Figure 10.

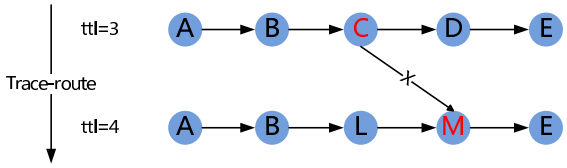

Fig. 10. Example for normal route shift, where the route changed during the two ping requests of the same traceroute.

It was important to notice in Figure10 that, both of the two routes reached destination E, with C on the third hop and M on the fourth hop. We identified this kind of links by finding valid route with the same router on the same hop. 45 out of the 80 records fell into this category.

Complex Route Shift

This was a more complicated situation than in the first case. There were three key differences comparing with the first category.

1) A complex route shift may shift among more than 2 routes within a traceroute.
2) The newly introduced route was allowed to be a route that did not reach the destination.
3) It was possible that none of other traceroutes record could match a node on a certain hop of a complex route shift.

Considering a traceroute with 5 hops, an example was shown in Figure 11.

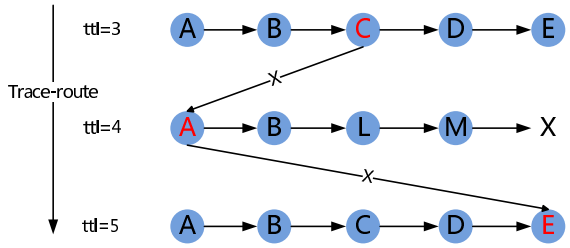

Fig. 11. Example for complex route shift we extracted from the traceroute data.

From Figure 11, it was hard to tell what exactly happened. Although we saw node A on hop 4, we could not find A again on the 4th hop from any other records. Hop 5 behaved similar with normal route shift. There were 9 records fall into this category in total.

Shortcut

Shortcut was a special case of a normal route shift, with one of the route in the shift to be an invalid route. An example is shown in figure:12.

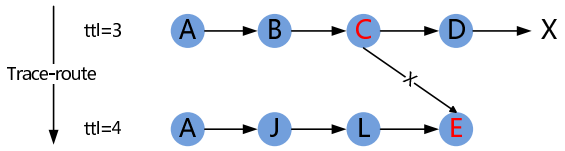

Fig. 12. Example for shortcut, that one of the routers on the route found and switched immediately to another route.

To explain this phenomenon in a more intuitive way, we could assume that one of the router is down, so traceroute could not get to the destination due to the router failure. Then, during the traceroute request, the router was fixed and went back to work and traceroute got to the destination immediately on the next hop. There were 3 records fell into this category.

Trapped in loop

This was another special case of normal route shift, where one of the route was a loopy one. An example was shown in figure:13.

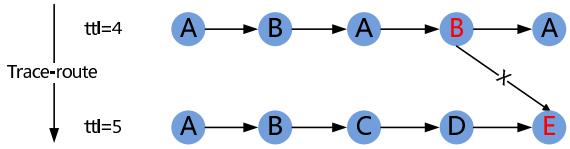

Fig. 13. Example for trapped in loop, where one of the router in the route was a loopy one.

There were 15 records in total fell into this category.

Cannot verified

The remained 8 records fell into this category. For these records, there existed some node that was never seen in other traceroute records with the same source and destination. Thus, there was no way we can tell what happened with such records.

In order to further analyze why these non-existent links occurred, we tried to find out the correlation between these links and the failures.

26 out of the 80 non-existent links happened at the same time as one failure somewhere in the network. But only 8 among these 26 records had one of its nonexistent link involved in the failure. Thus, generally speaking, network failures were not the main cause of nonexistent links. In addition, only 11 out of the 80 records with non-existent links failed to reach the destination.

In conclusion, non-existent links occurred in a very small fraction of data, and had relatively small impact on the performance of network, since most of them arrived at the destination eventually and many of them turned out to be normal routing changes of the network.

C. Failure Analysis

Given the traceroute data, we were able to take a closer look at the ISIS failures, which could potentially help us better understand the network behaviors and then try to reduce the number of failures. From all the 40073 failures we've ever pinged, only 125 (3.11%) of them had ever caused a route change from at least one source. This was a relatively small number comparing with the total failures we detected, which I thought is mainly because we only had 6 sources to send traceroute requests and there are only 3 of them worked as we expected, as well as most of the failures lasted for very short time period. It would be considerably helpful if we could have more sources sending traceroute to the failure end hosts, which would probably give us more interesting routing changes.

And about half of these changes, 63 out of 125 (50.40%), were not detectable because we could only get more or less unrecognized hops displayed as ** from traceroute. Since we couldn't recognize them, we didn't have their route weights and were not able to draw any conclusions on how their routes change quantitatively. But we could still do some analysis about the accessibility of the failure ends.

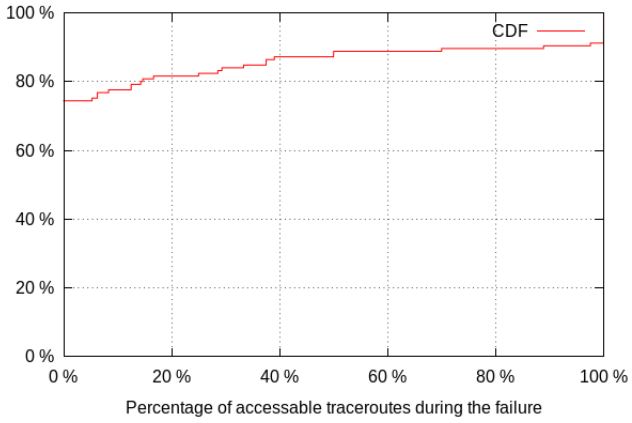

Fig. 14. Percentage CDF of accessible pings during the link failure time. 74.40% of the 125 failures which had caused route changes hadn't affected the accessibility of the end hosts, which meant the routing protocaol worked pretty well during those failures.

It was clear in Figure 14, that the end hosts of 93 out of 125 (74.40%) failures have kept unreachable during the whole failure period. Because all these 125 failures had caused route changes, which is to say that all these routes were affected by the failures somehow, we had to say that the routing protocols were performing relatively poor. However, there were still 8.80% of the failures had never affected the availability of the ends of the failure links during the whole failure period.

Furthermore, we did some statistics for the failure links, where a few links (192) occurred large number of failures (40073). From Figure 15, we could easily find that most of the failure links had small numbers of failures ever occurred. But the most surprising part of this figure is that there are some links that had more than thousands of failures during this period of time, which were supposed to be taken care of after tens of frequent occurrences.

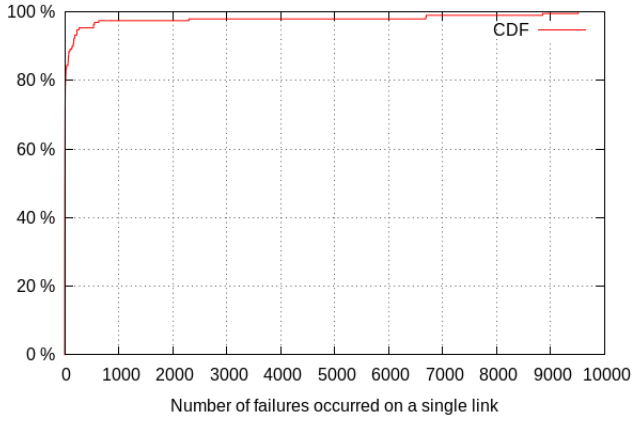

Fig. 15. The distribution CDF of the failure links. Most of the failure links had very small numbers of failures, where 74.48% of them had occurred less or equal 10 failures. However, there are some links had very frequent failures that were more than 8000 times during this period.

D. Routing Performance Characterization

With the help of the link weights, we also did a characterization on how well the routing protocols were doing on calculating the shortest routing paths. Among the 70831 traceroute records in total, we successfully extracted 52431 (74.02%) recognizable traceroutes. The rest of the traceroutes were unrecognizable because those router in CENIC did not support UDP traceroute, which meant we could only get ** as transmitting through these routers.

In Figure 16, it ware clear that most of the actual path weights were already the optimal ones, while there remained 7.64% were not. We would like to further analyze how bad those local optimums were, but it didn't make sense to compare the actual weights since the link weights were not consecutive at all.

We had also tried to carry out analysis about how the failures affected the network routes and their performance. In order to do that, we would need to understand the pattern manually before we could tell the program how to do so. The biggest

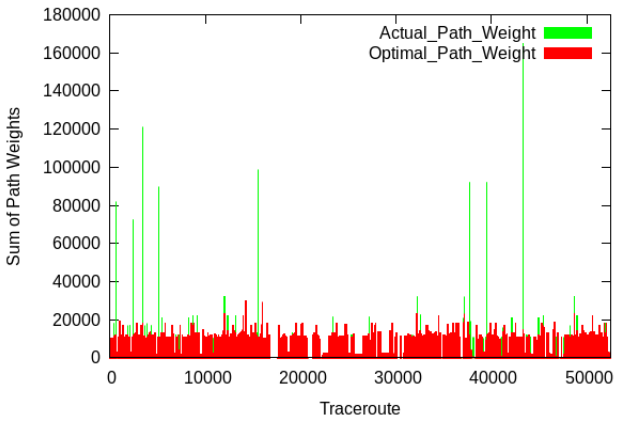

Fig. 16. The actual route weight compared with the optimal route weight calculating with the link weights. 48426 out of 52431 (92.36%) traceroutes successfully chose the global optimal route, which meant the routing protocols were working relatively well on this.

issue we encountered was to tell why the routing protocols responded like this and the correlation between routing changes and the failure links from thousands of records. Large amount of manual recognition and classification was inevitable while we were so limited on time, so we had to leave this to the future work.

VII. Conclusion

In this paper, we show an approach to extract the network failure information of CENIC with fairly cheap and commonly applicable mechanism without expensive probing infrastructures. Then we propose a methodology to analyze and characterize those extracted failures and detect the unexpected network behaviors including routing loops and non-existent links. And we also carry out a study on how the routing protocols are working on calculating the shortest paths. We believe our mechanism is fairly general and applicable for most of the common production networks.

Acknowledgment

We'd like to thank Prefessor Snoeren and Siva's advice on this project. And we do really appreciate the help from Dr Daniel Turner who provided most of the data and patiently answered every single question we asked. Unfortunately, this project is not funded by any fundation.

References

[1] Paul Baran. On distributed communications: I. introduction to distributed communications networks. *Volumes IXI RAND Corporation Research Documents August*, 12(1):51, 1964.

[2] CENIC. The corporation for education network initiatives in california. http://www.cenic.org/, March 2013.

[3] Abhishek Chandra, Rohini Prinja, Sourabh Jain, and ZhiLi Zhang. Co-designing the failure analysis and monitoring of large-scale systems. *SIGMETRICS Perform. Eval. Rev.*, 36(2):10–15, August 2008.

[4] Ítalo Cunha, Renata Teixeira, Darryl Veitch, and Christophe Diot. Predicting and tracking internet path changes. *SIGCOMM-Computer Communication Review*, 41(4):122, 2011.

[5] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, pages 350–361, New York, NY, USA, 2011. ACM.

[6] Craig Labovitz, Abha Ahuja, and Farnam Jahanian. Experimental study of internet stability and backbone failures. In *Fault-Tolerant Computing, 1999. Digest of Papers. Twenty-Ninth Annual International Symposium on*, pages 278–285. IEEE, 1999.

[7] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, Yashar Ganjali, and Christophe Diot. Characterization of failures in an operational ip backbone network. *IEEE/ACM Transactions on Networking (TON)*, 16(4):749–762, 2008.

[8] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, Yashar Ganjali, and Christophe Diot. Characterization of failures in an operational ip backbone network. *IEEE/ACM Trans. Netw.*, 16(4):749–762, August 2008.

[9] Venkata N Padmanabhan, Sriram Ramabhadran, Sharad Agarwal, and Jitendra Padhye. A study of end-to-end web access failures. In *Proceedings of the 2006 ACM CoNEXT conference*, page 15. ACM, 2006.

[10] Vern Paxson. End-to-end routing behavior in the internet. *Networking, IEEE/ACM Transactions on*, 5(5):601–615, 1997.

[11] Aman Shaikh, Chris Isett, Albert Greenberg, Matthew Roughan, and Joel Gottlieb. A case study of ospf behavior in a large enterprise network. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, IMW '02, pages 217–230, New York, NY, USA, 2002. ACM.

[12] Daniel Turner, Kirill Levchenko, Alex C Snoeren, and Stefan Savage. California fault lines: understanding the causes and impact of network failures. *ACM SIGCOMM Computer Communication Review*, 40(4):315–326, 2010.