Reproduction and Extension of Targeted Redundancy Dissemination Graphs

Zhiyong Chen
University of Pittsburgh
Department of Computer Science
Pittsburgh,PA
zhc79@pitt.edu

Shixiang Long
University of Pittsburgh
Department of Information Science
Pittsburgh,PA
shl199@pitt.edu

A. INTRODUCTION

Applications such as remote robotic surgery demands combinations of timeliness and reliability requirements in order to provide interaction between people in a natural way where round trip delay must be less than 130ms. However, the Internet only support reliable but not timely protocol such as TCP or timely with best-effort reliability protocol such as UDP.

The author develop a timely dissemination graph based routing approach consists of four pre-compute sub-graphs, and a fast problem detection system based on overlay network architecture. The approach will use two disjoint paths in normal case and switch to either source or destination or robust source-destination problem dissemination graph if a problem is detected by the system. In the original paper[2],the result shows that the target redundancy dissemination graph achieve nearly optimal performance, covering 99.81% gap between time constrained flooding and single path, and the approach achieve reasonable cost with approximately 2% cost increase than two disjoint paths.

We attempt to evaluate the performance - availability, reliability and cost of different approaches under real world network condition,since original paper show highly reliability with relatively small cost using target redundancy approach. And we plan to visually compare static two disjoint paths and target redundancy approach, in order to do so, we will collect received packets with their latency over two minutes. Since the result of original paper shows that even traditional single path would provide high reliability ,so it is worth to compare the reliability performance between target redundancy with other approaches under high loss rate network condition.The paper makes the assumption that bandwidth limit is not a concern and congestion control is always a hot topic in network therefore we would to see the reliability performance when large amount of congestion exist in overlay network.

Overall,we have three metrices to measure, availability, reliability and cost. Cost is the simplest one. Spines provide cost statistics periodically and we simply just collect cost from log file. Reliability means percentage of packet loss in total.In order to measure reliability, we count the number of lost packets and late packets and then calculate the reliability. Unavailability means the loss rate on a flow is above 50% over a period of time. To measure unavailability, we set fixed sending rate - 100 packets per second and set checkpoint equal to 100 in receiver,which check loss rate every 100 packets. If the loss rate is above 50%, the receiver will increase unavailable time by one.

After doing all the test, we find out that targeted redundancy approach could provide highly reliability and availability-covering 99.41% gap between optimal scheme and traditional single path. However, the cost of approach is 3 times as much as two disjoint paths. Through cost analysis, we find out that source problem and destination problem graphs are using all the time during the test. And after increasing the loss rate of links which includes source to 5, target redundancy approach will still provide high reliability, covering 99.43% gap between optimal scheme and single path approach. However, the performance under congestion network shows that target redundancy approach works even worse than traditional single path and the packet loss rate increases dramatically.

B. EXPERIMENTAL ENVIRONMENT

A. Set Up

In order to emulate overlay network in original paper, we deploy overlay nodes on different Geni sites. For the node which has corresponding geographical Geni site, we create a virtual machine with publicly routable IP on that site and run Spines. For example, author deploys overlay node in Chicago, we create a virtual machine at University of Chicago Geni site instead. For the node which does not have corresponding geographical Geni site, we simply deploy it at Case Western Geni site and use Spines setlink method with latency which we look up from professional website Wonder

Routing Approach	Availability (%)	Unavailability (seconds per flow per week)	Reliability (%)	Reliability (Packets lost per million)
Time-Constrained Flooding	99.97	181.44	99.98453	154.7
Targeted Redundancy	99.967	199.58	99.98153	184.7
Dynamic Two Disjoint Paths	99.93	423.36	99.95567	443.3
Static Two Disjoint Paths	99.9033	584.84	99.9479	521
Redundant Single Path	99.0433	5786	99.5226	4774
Single Path	98.99	6108.5	99.4785	5215

TABLE I: AGGREGATE AVAILABILITY AND RELIABILITY WITH 65MS LATENCY CONSTRAINT

Network [1] and constant loss rate 0.2% to emulate real world network condition. Besides,in order to make less messy, we write a program to automatically run Spines in the background.

B. Methods

We compare six different approaches under different network condition. Spines provide a easy way to implement flooding , disjoint paths and targeted redundancy graph by tune the parameter Spines disjoint path k. We use sp_bflooder.c as our sender and receiver prototype and modify it to meet our need. To simulate targeted redundancy graph, we set parameter k = 6. For dynamic single path and dynamic two disjoint paths, we set parameter k = to 1 and k= 2 respectively. To simulate static two paths approach, we only open a subgroup of overlay nodes,which are DFW,DEN,ATL and LAX and set k=2. To simulate time-constrained flooding approach, we turn off a subgroup of overlay nodes, which are NYC, HKG,FRA,LON since packets will not be able to reach destination through these nodes and then set k=0, which means flooding. To simulate redundant single path, we add one more send to method in sender code and set k = 1.

C. Performance Results

A. Reproduce Plan:Comparison of Approaches

Table I presents reliability and availability results of different approaches and the result of our experiment basically accords with the result of original paper[2] in terms of order of reliability and availability performance.All approaches can provide more than 99% reliability and targeted redundancy approach achieve all most the same good result as time constrained flooding approach.However,the number of packet loss per million and unavailable second per week per flow are much more higher than that of original paper.

From Table II,the results show that our dissemination graph approach with targeted redundancy achieves nearly optimal reliability, covering 99.41% of the gap between time-constrained flooding and traditional single path routing.And two disjoint paths offer a substantial improvement over a single path,covering about 94.3% of that gap between optimal scheme and single path approach.

Routing Approach	Overall(%)	Scaled Cost
Time-Constrained Flooding	100	11.012
Targeted Redundancy	99.41	5.858
Dynamic Two Disjoint Paths	94.3	2.000
Static Two Disjoint Paths	92.7	2.001
Redundant Single Path	8.71	2.000
Single Path	0	1

TABLE II: PERCENT OF THE BENEFIT OF TIME-CONSTRAINED FLOODING OBTAINED BY EACH APPROACH AND SCALED COST

Table II also presents scaled cost.Basically scaled cost of different approaches match the paper except for targeted redundancy. The scaled cost of targeted redundancy approach is 5.858 while in the table II of original paper[2], the scaled cost of targeted redundancy approach is 2.098 , the result is bigger than we expect.In order to know why it cost so much, we do the cost analysis.

B. Cost Analysis

From four dissemination subgraphs of original paper, it is easy to conclude that node SJC transmit packets if and only if when problem occurs around destination and node JHU transmit packets if and only if when problem occurs around source and node DFW transmit packets all the time. Here,robust source-destination problem graph could be considered as a intersection of source problem graph and destination problem graph. Let D represents total time, J represents time using source problem graph and S represents time using destination problem graph,then we could calculate the ratio between them:

$$D : J : S \approx \frac{cost(DFW)}{Rate} : \frac{cost(JHU)}{Rate} : \frac{cost(SJC)}{Rate} \tag{1}$$

The reason for using approximately equal to rather than equal to is that even though the cost includes the number of node resending packets,loss rate of links between nodes is only 0.2% and the goal is to get a approximately ratio, therefore we could ignore the time node resending packets. The following table show the count statistics.

Location	Cost
DFW	1004084
JHU	1001370.7
SJC	1003105.3

From the table, we can see the cost of DFW, JHU, SJC are almost same, which basically means the graph use source problem and destination problem graph all the time. Besides, we find geni network is very unstable. So, it is reasonable to reason this abnormal data cased by terrible network environment.

C. Reproduce Plan : Visual Comparison

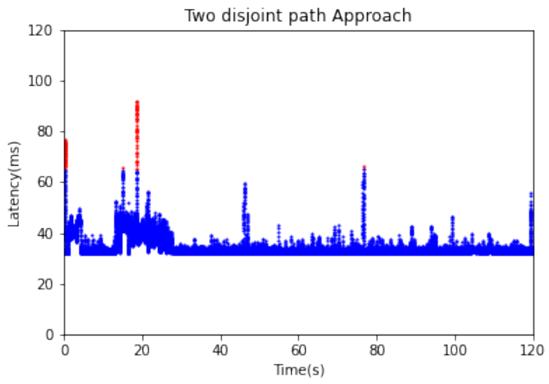

Fig. 1: Packets received and dropped over a 120-second interval from Atlanta to Los Angeles, using two node disjoint approach at the destination (105 lost or late packets, 0 packets with latency over 120ms not shown.)

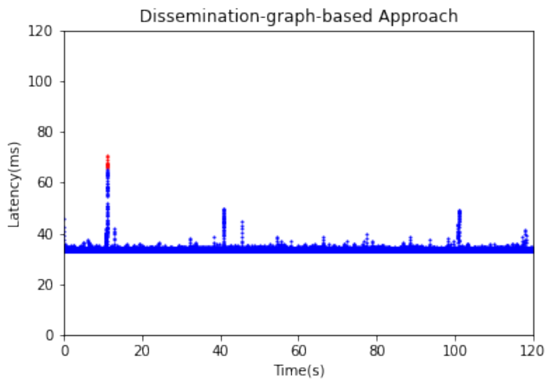

Fig. 2: Packets received and dropped over a 120-second interval from Atlanta to Los Angeles, using our novel dissemination-graph-based approach to add targeted redundancy at the destination (12 lost or late packets, 0 packets with latency over 120ms not shown.)

We visually compare static two disjoint paths and target redundancy approach. In order to do so, we modify the program by recording the latency and received time of every received packets including late packets into a file.Then we select a subgroup of packets whose received time is within two minutes of receiving the first packet and we plot latency and received time using them. The vertical lines indicate recovery processes- problems detected in one of the overlay nodes and network start hop by hop recovery, that is the reason why so many packets with different latency arrive at the same time and form a vertical line. The results show that 105 packets were lost using static two disjoint paths while only 12 packets were lost using targeted redundancy approach.

D. Extend Plan: High Loss Rate Network condition

Increase loss rate of all links containing source to 5			
Routing Approach	Reliability(%)	Packets lost per million	Overall (%)
Time-Constrained Flooding	98.95365	10463.5	100
Targeted Redundancy	98.92495	10750.5	99.43
Static Two Disjoint Paths	96.78145	32185.5	56.74
Dynamic Two Disjoint Paths	97.44185	25581.5	69.89
Single Path	93.9323	60677	0

TABLE III: High Loss rate network case 1

Increase loss rate of all links in static two paths to 5			
Routing Approach	Reliability(%)	Packets lost per million	Overall(%)
Time-Constrained Flooding	99.97985	201.5	100
Targeted Redundancy	99.96865	313.5	99.3
Static Two Disjoint Paths	98.3657	16343	0
Dynamic Two Disjoint Paths	99.92	800	96.29
Single Path	99.28235	7176.5	56.79

TABLE IV: High Loss rate network case 2

To compare the reliability performance between different approaches under different network condition with high loss rate,we consider two different network situations, the first one is that all links containing the source node have high loss rate. To create such network, we use set link method to increase loss rate of links ATL-JHU, ATL-WAS,ATL-CHI,ATL-DEN,ATL-DFW to 5.The targeted redundancy approach still works good and provide 98.9% reliability and covering 99.4% gap between optimal scheme and single path. The second network condition is that all links in static two paths have high loss rate. To create such network, we use setlink method to increase loss rate of links ATL-DEN,DEN-LAX,ATL-DFW,DFW-LAX to 5. Under this situation, static two disjoint paths behave worse than dynamic single path and targeted redundancy approach still works good,covering 99.3% gap between optimal scheme and static two disjoint paths. Use these two typical case, we believe targeted redundancy dissemination graph have strong resilience to high loss rate network condition.However given the cost analysis we did in reproduce part, we know that,in normal case,targeted redundancy approach will use source and destination approaches all the time therefore we cannot manage to figure out how much cost will increase when the redundancy graph approach is under high loss rate network.

E. Extend Plan: Congestion

Routing Approach	Sending Rate	Packets lost per million	Reliability%
Time-Constrained Flooding	5000	133	99.9867
Time-Constrained Flooding	10000	616770	38.323
Time-Constrained Flooding	15000	896380	10.362
Targeted Redundancy	5000	136	99.9864
Targeted Redundancy	10000	358450	64.155
Targeted Redundancy	15000	832701	16.7299
Two Disjoint Paths	5000	223	99.9777
Two Disjoint Paths	10000	1960	99.804
Two Disjoint Paths	15000	48536	95.1464
Single Path	5000	5031	99.4969
Single Path	10000	29363	97.0637
Single Path	15000	244887	75.5113

TABLE V: Congestion Impact

To find out the impact of congestion on performance with different routing approaches, we set sending rate equal to 5000 kbps,10000 kbps,15000 kbps respectively and send one million packet using different routing approaches. Table V shows the result.When sending rate equal to 5000 kbps, there is no congestion in overlay network, therefore the targeted redundancy approach provide good reliability. However, when sending rate is equal to or more than 10000 kbps, which means lots of congestion in overlay network, targeted redundancy dissemination graph behaves even worse than single path and two disjoint paths provide relatively good reliability performance. And about 83% of packets are lost using targeted redundancy approach when sending rate is 15000kbps.The possible explanation might be when congestion happens, overlay nodes drop packets and the system detect the loss and switch to source or destination problem graph which makes situation worse, since it will cause more congestion and loss rate immediately increase dramatically.

D. Discussion and Future Work

We find several limitation of our experiment.The first limitation is we can not create a stable network even though we attempt to deploy all 12 overlay nodes in one Geni site.During the test,spines,via terminal, shows that we have lots of network problems.The test result is very bad at the beginning - loss rate of single path was higher than 10%.However,when I switch testing time from daytime to early morning, the situation become much better- the loss rate and availability behave only slightly different that of the original paper.But through cost analysis, targeted redundancy graph actually use source and destination problem graph all the time. Given this situation, some initial plan will not be done.For example, I plan to see how much cost will increase when we use targeted redundancy graph under high loss network condition.If the target redundancy graph use source or destination graph all the time even at normal case,there is no difference with using it under high loss network case. The second limitation is clock synchronization.During the small test, we record the latency of packets.Sometimes,we find that the latency of some received packets are negative,indicating there exists clock skew.We just makes assumption that the clock is synchronized, but the problem do exists. In one of our recorded data, the latency of all packets received are around 65ms, which must be caused due to clock skew.

As we state before, the targeted redundancy have terrible performance under congestion, therefore adding congestion control is a necessity.In overlay network,we could monitor the bandwidth,which is a strategy for us.I come up with a basic solution, when congestion happens or there is limited bandwidth, the targeted redundancy approach could switch to dynamic two disjoint paths rather than source or destination problem graph since dynamic two disjoint paths perform relatively good.If we have more time, we would like to see its performance. In addition, we would like to add clock skew calculation program between spines.But it may not be too complicated, since it is the responsibility of receiver and sender to adjust the clock.

E. Conclusion

We present the reproduction results of dissemination graphs under emulated real world network condition and analysis the cost.Besides, we evaluate the performance of dissemination under two different high loss network and more congestion network. We also demonstrated that this approach can cover over 99% of the performance gap between a traditional single path approach and an optimal but with almost 5 times cost.To make dissemination graph robust to complex network situations, we might add congestion control and clock synchronization tools.

References

[1] WonderNetwork, https://wondernetwork.com/.

[2] Babay, Amy, et al. “Timely, Reliable, and Cost-Effective Internet Transport Service Using Dissemination Graphs.” 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, https://doi.org/10.1109/icdcs.2017.63.

F. Appendix

A. Configuration

The first thing we do is to create 12 virtual machines with publicly routable IP on GENI. At UCLA InstaGENI site, we create one virtual machine to represent LAX. At University of Texas InstaGENI site, we create two virtual machines to represent DFW and DEN respectively. At University of Chicago InstaGENI site, we create one virtual machine to represent CHI. At Georgia Tech InstaGENI site, we create one virtual machine to represent ATL. At NYU InstaGENI site, we create one virtual machine to represent NYC. At University of California -San Diego Insta GENI site, we create one virtual machine to represent SJC. At University of Washington InstaGENI, we create one virtual machine to represent WAS and JHU respectively. At Case Western InstaGENI, we create three virtual machines to represent FRA, LON and HKG respectively. Then we look up the IP address of each node and finish Host part of configuration in spines.conf under daemon folder. You could find our host configuration in spines.conf file. For the node have a corresponding Geni site, we use ping command to get round trip time between two nodes ,divide it by two and round it to closest integer.For the node which does not have a corresponding Geni site, we look up round trip time from a professional network website, wondernetwork [1],which allow you to get latency between two cities.We take the average of ping time, divide it by two and round it to closest integer. After doing so, we should be able to fill the Edges Part of configuration. You could find our edge configuration in spines.conf file. After finishing spines configuration, we will run spines through command ./spines -p 8200 -m -w problem -pc

To make life easier, we make a small script deploySpines_exceptRec.ipynb to automatically update and deploy the spines overlay network on 12 virtual machines.The script using python3 paramiko module, control remote 12 virtual machines. The module read the username,private key and ip from the files, use ssh to send the command to remote virtual machines. For deployment: In order to allow spines run in background, we use ”nohup”.For update: the code of every virtual machines are pulled from our repository Everytime we run the spines, the script would update the code of all virtual machines. In this case, we only need to push our new change to our repository, then all of the code can be updated.

Then use set link with latency we get before and loss rate which is commonly considered as 0.2% to emulate spines overlay network under real world network condition.You could find setlink instruction in Setlink.pdf.

B. Loss number, cost and packet latency

For different approaches, we all send 3 million packets three times and directly get number of lost packets and loss rate from terminal. After testing, we directly collect cost from spines terminal and sum them up for total cost. In order to do so, we modify sp_bflooder c program and rename it to latency_traffic to match our need. For the user want to replicate our process, just respectively run the instructions in Instruction.pdf. To be careful, when testing static two paths approach, we only open a subgroup of overlay nodes, which are DFW, DEN, ATL and LAX. When testing time-constrained flooding approach, we turn off a subgroup of overlay nodes, which are NYC, HKG, FRA and LON since packets will not be able to reach destination through these nodes. To test redundant single path, we add one more send to method in sender code.

To visually show the relation between latency and arriving time, we modify latency_traffic, recording every received packets with arriving time and latency and after running the program, the record will be stored in to latency.txt. And you could you use scp command to copy it from remote machine to your local computer and we write a python program Fig6 and Fig7.ipynb to show the relation.

C. Availability

In order to measure availability, we modify the sp_bflooder c program and rename it to availability_traffic. The program will send one packet per 10 ms, which is equal to 100 packets per second. We basically set check point equal to 100. So every 100 packets we will trace loss rate, if loss rate exceeds 50%, we will consider the last second is unavailability. In the end of program, it will calculate the total time as the difference between the end of program and the time when we receive the first packet, then we will get the unavailability time and availability. For easy test, the instruction will be in Instruction.pdf.

D. Extend Part: High loss rate network cases

The first thing to do is set up high loss rate network environment, in order to do so, respectively use the setlink methods in the high loss rate network case 1 and 2 in setlink.pdf. After setting up, we run the latency_traffic c program to get number of loss packets.

E. Extend Part2: More congestion

The setup is the same as the reproduce part, but this time, we run latency_traffic.c program with different R – 5000,10000,15000. Then we get the number of loss packets.

F. Calculation of Unavailability

In the original paper, the unit of Unavailability is seconds per flow per week. But in our test,we get unavailability time per million packets and sending million packets need 10000s.We scale it to the unit the original paper use simply by multiplying 60.48.