IDENTIFYING LINK FAILUERS

VADAPALLI MADHAVI VENKATA VISALAKSHI

K R RAJESWARI

B.V. Raju College, Vishunupur::Bhimavaram

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COLLEGE OF ENGINEERING ADIKAVI NANNAYA, UNIVERSITY RAJAHMAHENDRAVARAM

1.INTRODUCTION

In this project a formal approach for analyzing routing data to identify the origin of routing changes is provided. Using a simplified model of BGP and a graph model of the Internet, FindChange algorithm set is presented. Note that even a single link failure or link addition can result in large number of BGP path changes, but finding this change without knowledge of the underlying topology can be a challenge. As a algorithms for understanding internet route changes, identifying the single link failure or addition that caused an observed set of changes is focused. This approach takes snapshots of routing tables collected from vantage points, at two different time instances, and without knowing the underlying topology, locates the link that initiated the routing changes, as precisely as possible.

1.1 Distinctions between BGP monitoring and this model

The general problem of understanding BGP behavior based on observed updates is an open challenge. As a first step, a few simplifying assumptions are made in this formal model. The edge between two nodes in VN is modeled as a single link EN and assume this link is either available or has failed. But in actual BGP operations, the link between two AS can be many physical connections. For example, networks of large ISPs are connected at many places (packets can be exchanged at any of these connection points) and only some of these many physical connections are likely to fail at once. An AS is modeled as a node in VN and assume each node has one “best” path. In practice, a large AS node is not a single atomic entity and different contiguous portions of the AS, may select and advertise different best paths. Despite these simplifications, the formal model still presents an interesting challenge. The techniques used to find faults in this model

can be applied to the actual BGP monitoring data, taking the above assumptions into consideration when reviewing the results.

1.2 Identifying Faults Using a Single Vantage Point

Considering only the view from a single monitoring point M, and an algorithm is provided that gives a possible explanation for the routing changes observed at M. More formally, two path trees rooted at M: T0 = (V0, E0)is the path tree at time to and T1 = (V1, E1) is the path tree at time t1are given. Both T0andT1 were computed in some unspecified graph G = (V, E) and if T0≠ T1,some link(s) in G = (V, E) must have failed (or recovered). Objective of this model is to identify some scenario of failed (and/or recovered) links that explain the change from T0to T1. Toward this end, an algorithm that assigns labels to each edge and identifies one possible scenario for the route change event is presented. Later it is shown how the output of this algorithm can be easily extended to identify every possible single-event scenario and how to use the view from single monitoring point to identify, as precisely as possible, the failed (or recovered) link.

Algorithm Find Change takes trees T0 = (V0, E0) and T1 = (V1, E1) as input and labels each edge in E0U E1 as either unchanged, vanished, or appeared. A vanished link is present in T0, but not present in T1. This can occur due to the failure of the link or the link may vanish as a consequence of some other change. The actual failed edges are distinguished from the other vanished edges by labeling failed edges as failed. Similarly, edges that recovered or whose additions caused route changes are marked added. Note that edges marked appeared may not have changed state (from down to up), but simply became part of the path tree as a consequence of some other event.

Conceptually, the algorithm is relatively simple. To identify a link failure, the algorithm combines T0and T1 and then starts by calculating a path tree, Tfail in this combined graph. Note that if only a link failure has occurred, Tfail=T0and the algorithm progressively transforms Tfail by removing links that are absent in T1.The failed links are those whose removal changes Tfail and we adjust Tfail after each change. Similarly in the case of a link recovery, Tadd= T1 is transformed to get T0, to identify the restored link. It is not given whether a link failure or link recovery (or both) have occurred, but it can be observed that the two steps can be run in parallel as shown in Algorithm.

ALGORITHM: FindChange(T0, T1)

Input:T0 = (V0, E0) : The PT from M at t0 ; T1 = (V1, E1) ; The PT from M at t1 ;

Output: Marked edges: unchanged,

vanished, failed, added ; Let V = V0 U V1,

Eadd = Efail = E = E0 U E1 ;

Let Tfail = Tadd= PT(V,E) ;

For each e ϵ E do

if e ϵ E0 ∩ E1 then e = unchanged;

else if e ϵ E0 then e = vanished

if e ϵ Tfail then e = failed;

Efail = Efail – e;

Tfail = PT(V, Efail); else if e ϵ E1 then

e = appeared; if e ϵ Tadd then e = added;

Eadd = Eadd – e;

Tadd = PT(V, Eadd);

Figure below provides an example showing the execution of the algorithm. Three prefixes, P1,P2 and P3are advertised from AS-4, AS-6 and AS-5 (respectively). Figure 1 shows T1, the path tree at time t1. Note that the path to prefix P3 has not changed, but an event has changed the paths from M to prefixes P1 and P2. The algorithm will label each edge as unchanged, vanished, or appeared and will select one edge as the cause.

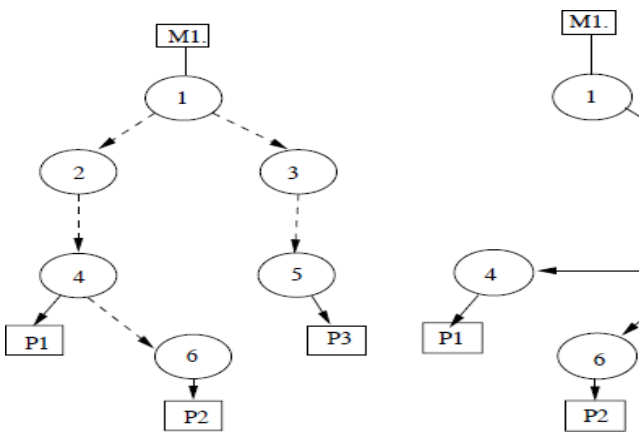


Figure 1: Input Graphs

The algorithm first combines the two trees T0 and T1 to obtain the graph shown in Figure 2 and computes a new path tree in this combined graph. Each edge is tested and labeled accordingly. Edge (M, 1) appears in both graphs and is labeled unchanged. Edge (1, 2) appears only in T0 and is labeled as vanished. Furthermore, edge (1,2) is in the PT Tfail and thus edge (1, 2) is marked as a failed link and anew PT Tfailis computed after excluding edge (1, 2), as shown in Figure 3. Each of the additional edges are tested and are either present in both trees and labeled as unchanged or not present in Tfail(or Tadd) and labeled as vanished or appeared. Note that for edges (5, 4) and (5, 6), since they are present in T1, their membership has to be checked in Tadd and not Tfail. The resulting labels are shown in Figure 4, where F denotes failed, A denotes appeared, V denotes vanished, and U denotes unchanged.

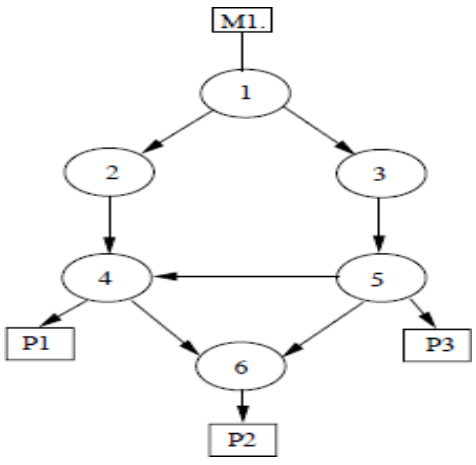


Figure 2: Graph after merging

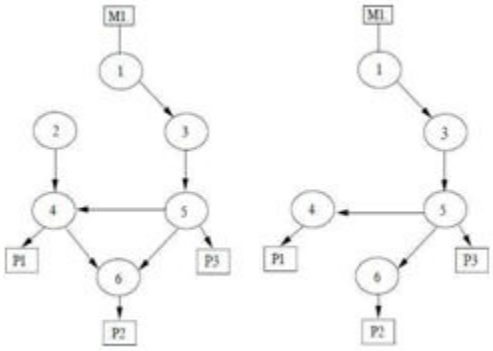


Figure 3: Removing failed edges

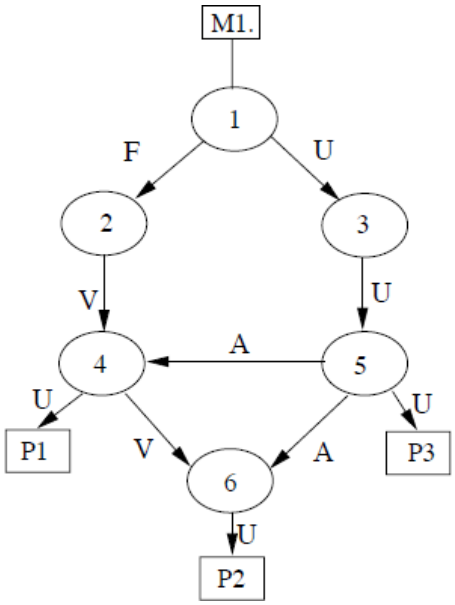


Figure 4: Labeling edges

In this figure

'F' represents failed 'U' represents unchanged 'A' represents available

Here, the algorithm identified link (1, 2) as failed link. All other links did not change state, though they moved into or out of the path tree as a consequence of this single failure. While this scenario is a feasible explanation for the change observed at M, it is not the only possible explanation. The failure of only link (2, 4) could have also caused the change fromT0 toT1. In addition, a variety of multi- failure/recovery events could have caused the change, such as the recovery of link (4, 5) combined with failure of links (4, 6) and link (2, 4). Any one of these events is a plausible explanation and the exact cause cannot be determined from only T0 and T1.

II. RELATED WORKS

This system provides an approach to identify link failures along with links that are restored and the links unchanged by implementing an algorithm called Find change using graph theory of internet.

Here, two snapshots of a network is taken as input in the form of graphs, at two different timings with each vertex indicating a system and each edge indicating link between two systems.

These two graphs are compared and a new graph is formed by merging these two graphs. Now each edge in the merged graph is taken and checked if it is present in both the graphs which have been taken as input, if yes then label that edge as an unchanged edge, else if the edge is present in the first graph then label it as failed link, else if the edge is present in second graph then label as restored link. Finally, the links which are failed, restored and unchanged are displayed.

III. PROBLEM STATEMENT

An ad-hoc combination of intuition,

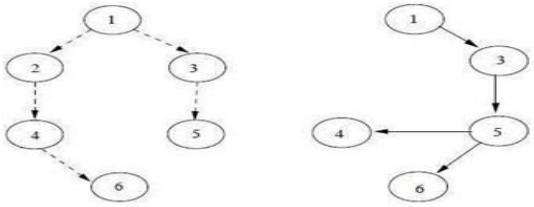
experience, and informal analysis is often used to speculate the causes of large swings in BGP updates. In addition to the challenge of scale, the underlying Internet topology is not known precisely and the monitoring points provide views from only a limited set of geographic locations. In such an environment, ad-hoc techniques are limited by the expertise of the administrator and it is not easy to identify the underlying events that cause BGP changes. To provide dependable global data delivery, analysis tools are needed that can help us to understand the BGP system and pinpoint the exact cause of connectivity changes.

IV PROPOSED SYSTEM

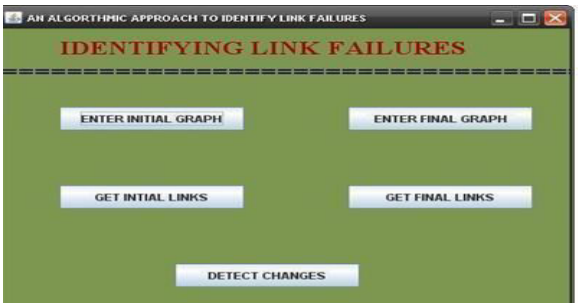
A formal approach is provided to identify the link failures which are supposed to cause route changes and pinpoint exact connectivity changes. Along with the links failed, the links which are restored or newly added, the links which remains unchanged are identified and the number of links failed, restored, unchanged.

V, RESULT ANALYSIS

Input graphs:

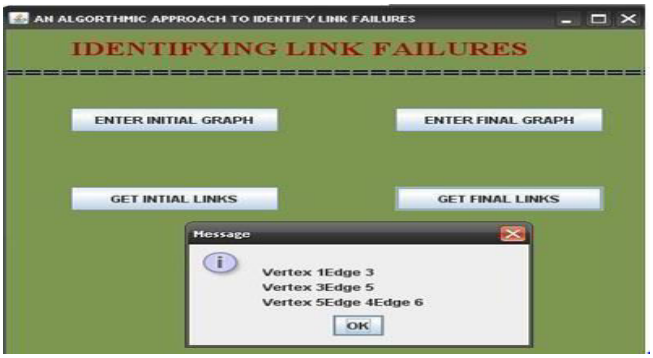


User interface:



Taking initial input:

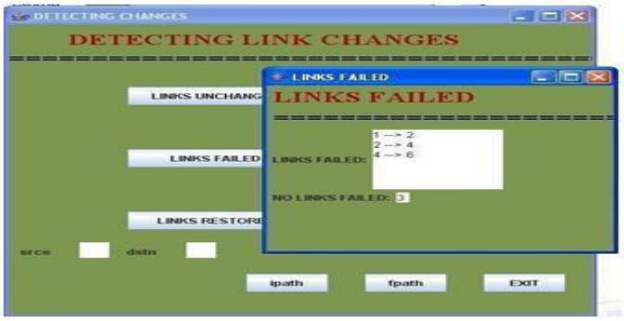
Displaying links in initial graph:



Displaying links in final graph:

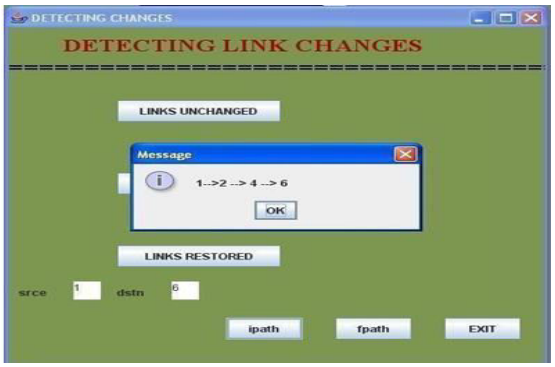
Showing links unchanged:

Showing links failed:

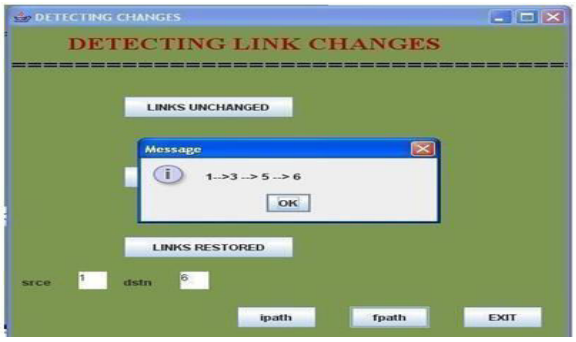


Showing links restored

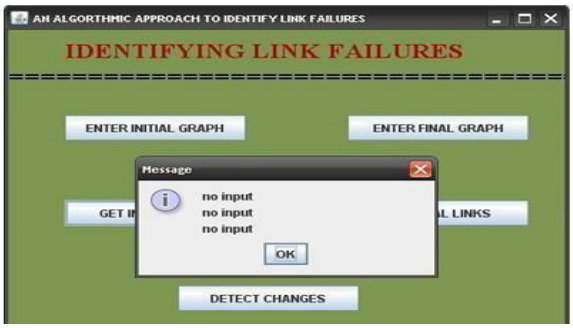
Path in initial graph



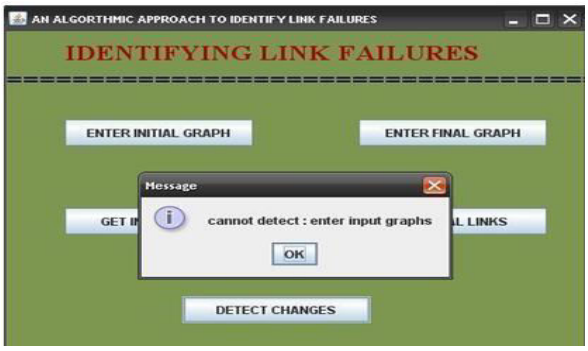
Path in final graph:



Input error:



Error Detecting Changes



VI. CONCLUSION

Utilizing the information collected from Single vantage points, Find Change has pin down the exact location of the connectivity changes. Connectivity changes such as link failures and link addition or restoration have been identified. In order to build diagnostic tools for the global routing infrastructure, FindChange algorithm is used to identify the origin of routing changes caused by a single link Failure. FindChange represents an important first step towards

applying formal methods to inter- domain routing diagnosis.

VII. FUTURE SCOPE

Scope for future work includes extending FindChange to apply over information collected from multiple vantage points and also to cover increasingly more complex failure cases.

VIII. BIBLIOGRAPHY

Book references:

1. "The Complete Reference JAVA 2" by Patrick Naughton and Herbert Schildt
2. "Computer Networks" by Andrew.S.Tanenbaum
3. "JAVA: How To Program" by Dietel and Deitel
4. "Software Engineering, A Practitioner's Approach" by Roger.S.Pressman
5. "Software Engineering" by Sommerville
6. "The Unified Modeling Language User Guide" by Grady Booch, James Rambaugh, Ivar Jacobson

Web references:

1. [Swings]

http://java.sun.com/docs/books/tutorial/uiswing/components/index.html

2. [Object oriented modeling and design]
http://www.visualcase.com/tutorials/uml-tutorial.htm

3.[Graph theory of internet]

http://expertvoices.nsdl.org/cornell-info204/2008/03/02/graph-theory-in-internet-hardware/

Paper references:

1. R. Govindan and H. Tangmunarun kit. Heuristics for internet map discovery. In IEEE INFOCOM 2000, pages 1471–1380, Tel Aviv, Israel, March 2000. IEEE.
2. Y. Rekhter and T. Li. A border gateway protocol (BGP-4). Request for Comment (RFC): 1771, Mar. 1995.
3. M. Lad, B. Zhang, X. Zhao, D. Massey, and L. Zhang. Analysis of BGP Update Surge during Slammer attack. In Proceedings of 5th International Workshop on Distributed Computing, Dec. 2003.