Automatic Discovery of Relationships Across Multiple Network Layers

Andrew Turner
ECE Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
andrewtu@cmu.edu

Hyong. S. Kim
ECE Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
hskim@cmu.edu

Tina Wong
ECE Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
tinawong@cmu.edu

ABSTRACT

As networks become increasingly large and complex the relationships and dependencies between network elements also grow in complexity and size. If an e-mail server goes off-line for ten minutes but never goes off-line again, it might not be a serious problem, and can be forgotten about. But if the e-mail server repeatedly goes off-line, the cause of the problem must be found. The difficulty for network operators is discovering what causes the problem – we address this issue and formalize a method to make event logs more useful to network operators.

In this paper we describe how network events can be correlated across multiple network layers, and utilize the temporal and spatial aspects of the event data to more accurately correlate network events than using the event descriptions alone. Our results show that by statistically analyzing Syslog data, a relationship graph can be automatically generated that shows relationships between network elements. We then go on to discuss how such a relationship graph, in combination with event correlation, can help operators more accurately discover the root cause of problems, and identify hidden relationships and dependencies within their networks.

Categories and Subject Descriptors:

C.2.3 **[Network Operations]**: Network monitoring

General Terms:

Management, Measurement, Reliability

Keywords:

Network Management, Root Cause Analysis, Probability, Sequential Pattern Mining, Syslog

INM'07, August 27--31, 2007, Kyoto, Japan.

1. INTRODUCTION

Modern communication networks continuously monitor and gather large amounts of data regarding the current state of the network; but large amounts of data do not necessarily mean large amounts of information. The information contained within the data must be discovered and extracted to become useful to the network's operators.

To better discover information contained within the data being collected, we propose a framework, called a 'relationship graph', that shows relationships between discrete network elements. To accomplish this, data from multiple network layers are correlated by event time, and then analyzed to discover sequential patterns of events. For example, event messages and errors from the application layer, transport layer, network layer and link layer can be combined and analyzed as if it all came from a single system; which they do, they all come from 'the network'. Using these patterns, the locations of a series of events matching a pattern can be recorded then analyzed to discover discrete network elements that frequently report sequential events in a pattern. The relationship graph shows the degree of belief we have that locations will report events in series, and does not necessarily mean that the elements are dependant on one another. For example, one element may report that a peer is down, while another element may then report that it is trying to reconnect to that peer. An operator viewing the events would want these two events correlated as they are related; the relationship graph will show a relationship between the two elements and help us to correlate the two events. The possible relationships between network elements that we have identified are:

- Physical - A is linked directly to B
- Common neighbor - A and B are both physically linked to C
- Hierarchal - A and B connect logically over the IP layer, but indirectly over a physical link
- Policy - A and B are at either end of an Label Switched Path
- Functional - Clients cannot access an e-mail server if the DNS server is down as they cannot resolve the host name

Using our method we can automatically discover all of the identified relationships between network elements, given that the elements have previously failed or reported network related events. This can help us better correlate future network errors and events, and also confirm or inform network operators of the relationships that are present within their networks. As our method is automatic and uses readily available network data, the

results of our method come at zero cost to the network operator. However, our method is not yet proficient at automatically classifying the relationship type between two network elements. In addition to this, the preliminary version of our method described in this paper uses Syslog, rather than event messages from multiple network layers, as its data source.

2. RELATED WORK

Many previous works have studied event correlation and sequential pattern mining, such as [2, 3, 4, 5, 6, 7, 8] to name but a few. Many papers have also used temporal heuristics to improve the accuracy of their results. However, while many papers have shown how these heuristics can be used to improve event correlation, few have created a useable system that could be widely adopted by network operators.

Previous research into the temporal aspect of event correlation has been discussed in [9, 10, 11]. The most basic temporal heuristic is that the next event in a sequence will occur within n seconds. More complex temporal methods use Hidden Markov Models or Dynamic Bayesian Belief Networks to more accurately correlate events. We choose to use a probability distribution for the timing between events, as events will be assigned to a pattern with a certain probability if multiple patterns that the event could belong to occur during the same time period; this is discussed further in Section 4.3.

[10, 12] both discuss using the topological information of events to more accurately correlate events. We use a method similar to [12], using empirical data to automatically create a topology of related network elements; this is discussed further in Section 4.4.

[13] describes Leslie graphs. Our relationship graph is similar to a Leslie graph in that it describes relationships between network elements, however, our method is concerned with using cross-layer log data to discover relationships rather than flow data. In short, our method is focused more on finding error relationships between servers, routers, switches, DWDMs, etc., rather than finding functional relationships between clients, protocols, servers, software, etc.; however, both approaches could presumably be applied to any networked system with some success.

3. BACKGROUND

Currently, due to non-standardized event logging systems, most data collected across multiple network layers is not correlated. This can often lead to the blame game; where operators from different network layers blame each other for problems, but it is extremely difficult to find the real cause. The End-to-end Diagnostic Discovery (EDDY) system was developed to address such issues[1]. EDDY allows events from multiple systems, multiple network layers, and multiple formats, to be processed and manipulated under a common framework. With this view of the network as a whole, rather than only viewing each individual network element or layer, EDDY is expected to provide much greater levels of problem diagnosis than current event logging and analysis systems. We plan to incorporate EDDY in our analysis in near future work.

[1] http://www.cmu.edu/computing/eddy/

The current implementation of our method uses Syslog data as a basis to discover relationships between network routers. The Syslog data is stored as a text file containing sequential events logged by the network routers.

- 2006-15-10T00:04:30|local6|err|int1|.pm3392.0.1. R64B66B: Loss of signal from the optics

Above is an example Syslog message, where at 4:30am on the 15th Oct 2006 the interface 'int1' reported loss of signal from optics. The Syslog text files contain no explicit data that describes the topology of the network they come from, they only contain multiple entries such as the example above.

To discover patterns of events in our network data, we use the Apriori algorithm [1]. The algorithm iterates over the event messages, looking for common patterns of events that meet a certain minimum confidence value. For example, if event B occurs 99% of the time after event A, event C occurs 97% of the time after event B, we would have the pattern A then B then C. We do not assume that A causes B, or that B causes C, only that pattern A then B then C occurs a statistically significant number of the times.

3.1 The Abilene network

During the initial phase of this work, the Abilene network was analyzed to confirm the feasibility of finding patterns within sequences of event messages. Throughout 2006, the Abilene network recorded over 11 million Syslog events, averaging 32,000 events per day. However, none of these events were in the highest three severity categories of Emergency, Alert or Critical. Figure 1 shows the number of events recorded each day for the period April – December 2006; the legend shows the event severity as defined by Cisco[2] and Juniper[3], where a lower number corresponds to a more severe event.

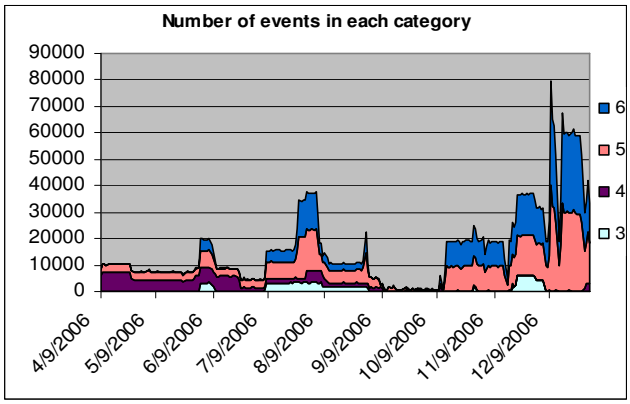

Figure 1: Events from the Abilene network

It can be seen from Figure 1 that spikes in the number of severity 4, 5 and 6 events occur when there are severity 3 events present. Using numeric data we discover that there are on average 8.6% more severity 4, 5 and 6 events on days when one or more

[2] http://www.cisco.com/warp/public/477/RME/rme_syslog.html

[3] http://www.juniper.net/techpubs/software/erx/junose72/swconfig-system-logs/download/logging-overview.pdf

severity 3 events occur. On days where 10 or more severity 3 errors occur, there are on average 22.4% more severity 4, 5 and 6 events. This confirms the intuitive assumption that when more-severe events occur, a greater number of less-severe events also occur. Therefore, we should be able to discover which event messages cause other event messages.

4. RELATIONSHIP DISCOVERY

4.1 Overview

Our method discovers patterns in sequences of events and uses the temporal information of the events to help remove false positive results. We then use the spatial information of the events to create a graph of related network elements, which can then also be used to remove false positive and false negative results, and can show network operators relationships that are present within their networks.

4.2 Pattern Discovery

As previously mentioned, to discover patterns of events in the Syslog data, we use the Apriori algorithm. While the Apriori algorithm is not the most time efficient algorithm to discover sequential association rules, it is the simplest. As this paper is concerned with finding relationships between network elements and confirming the validity of our methods – and is not currently required to run in real-time – the simplest and most transparent method is chosen. Currently, our method can process roughly 400 events per minute. Using the Apriori algorithm also allows us to easily add our own heuristics to the pattern discovery phase. As it is not feasible to analyze the entire history of a network in a single pass of the data, the data is split into discrete sections and processed; the discovered patterns are then combined to create a single pattern set.

4.3 Pattern Timing

To help increase the accuracy of the element relationship discovery, and to more accurately correlate related events, we use the timing between events as a heuristic in our model.

Rather than assuming that events are related if they occur within a short period of time from each other, we find a probability distribution for the time between each event occurrence in each pattern. This should make the event correlation more accurate in two ways. Firstly, as with the previously discussed temporal correlation methods, we can use the timing information to discount instances of patterns where the pattern of events may have happened by coincidence. For example, if we have the pattern A then B then C, and we have a 99% probability that B will occur within 5 seconds of A, we could discount an occurrence of the pattern if the time between A and B was 50 seconds; even though the events in the log file are in the order A then B then C.

The second way a probability distribution for the time difference between events makes the event correlation more accurate is when multiple patterns are interlaced. Figures 2 and 3 show an example of two interlaced patterns; A then B then C, and D then C then E. In Figures 2 and 3 the timing of the events in a pattern is simple; events occur 2 time intervals after the previous event in the pattern. If we generalized the timing information to, 'the next event occurs within 2 seconds', or we only used a closest distance match, we would incorrectly correlate event C at t_5 with event D at t_4; as shown in Figure 2. However, by using the more accurate timing distribution that the events occur exactly 2 time intervals after the previous event in the pattern, we can correctly correlate event C at t_5 with event B at t_3, and event C at t_6 and event D at t_4; as shown in Figure 3. In the general case, where the timing information is not so precise, events that could belong to multiple patterns are assigned to the pattern that they most probably belong to according to the timing and spatial heuristics.

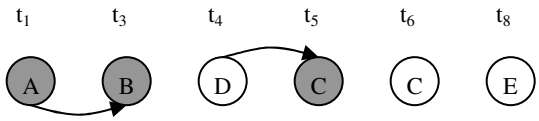

Figure 2. Incorrectly correlated interlaced patterns

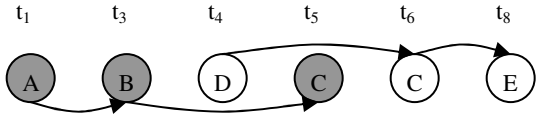

Figure 3. Correctly correlated interlaced patterns

Having accurate timing information becomes more important as additional network layers are added to the event correlation system – such as the situation when using the EDDY system. This is because of an increased probability of multiple event patterns occurring during overlapping time intervals – due to an overall increase in the number of events that will be present in the event correlation system. In addition to this, it is also more probable that events in a pattern will occur over a varied time period; for example, a DWDM will log an optical link cut instantly as it will notice the loss of light, but an e-mail server that reports errors due to that link cut may take 30 seconds to report the link is down – as it will wait for a time out period and try several reconnect attempts before logging that hosts are no longer reachable.

4.4 Event Locations

To create the relationship graph for the network elements, we calculate the probability that event e_2 occurred in location l_2 given that event e_1 occurred in location l_1 for each pattern; a fictional example is shown in Figure 3. In our current implementation a location is a single router, however, depending on the granularity of the event logging a location could be a component within a network element, such as a line card or hard disk.

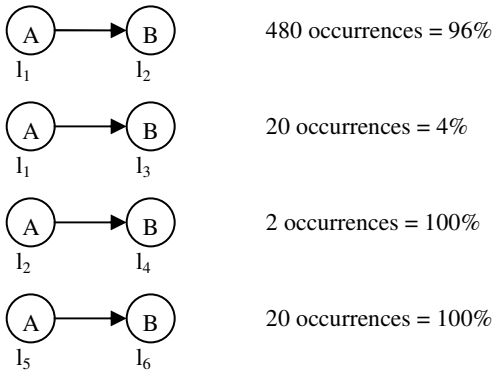

Figure 3. Event location occurrence

Using the spatial probability calculation for each pattern we create graphs of locations that we believe produce events in a sequential pattern, i.e. the locations are related. To create a generalized relationship graph, where we use the combined spatial probabilities from every pattern, we add heuristics to prevent coincidental occurrences or weak patterns skewing or adding noise to the spatial relationships.

The first heuristic we use when deciding whether to include a pattern's spatial probability in our calculations is the number of times the pattern is discovered during the pattern discovery phase. As previously mentioned, when the pattern discovery is performed the data is split into discrete sections; this gives us the number of times a pattern is discovered. For example, on one day there may be a specific pattern that occurs 10,000 times but occurs on no other day, another pattern may occur 100 times a day over a 100 day period. The pattern that occurs over the 100 day period is more useful to us, as the pattern that occurred on only one day could have been caused by a misconfiguration or error which was subsequently fixed, so the pattern will not occur again; whereas the pattern that occurs over 100 days is more persistently occurring, so the relationships that cause it are more likely to still be in place and valid. This is discussed further in Section 7.

The second heuristic we use is the total number of events that occur at each location. For example, in Figure 3, if location l_1 produces 10,000 events in total and location l_5 produces 1,000 events in total, we will factor up the number of occurrences of A then B at location l_5 then l_6 to compensate for the greater number of events that l_1 creates. This helps prevent relationships of 'quiet' network elements being obscured by 'noisy' network elements. During our experimentation, one router in our analysis produced 50% of the event messages. Without the above heuristic it seemed that the router was related to every other network element in our analysis, whereas in reality it produced a large number of messages due to communication with an external router that was not included in our analysis.

The third heuristic we use is the number of occurrences of each pattern for each location pair. For example, in Figure 3, l_2 then l_4 occurs 100% of the time when event A occurs at l_2. However, this only happens on 2 occasions, where as l_1 then l_2 occurs 96% of the time, but occurs on 480 occasions. Because of this, we set a minimum limit of the number of times a pattern must occur before we use it in our calculations – this value could be a fixed value or a minimum percentage of the total number of times the pattern has occurred. This heuristic is to prevent coincidental occurrences of events creating incorrect relationships between network elements.

Table 1. Combined spatial data (* using a 90% min cutoff)

	P_1	P_2	P_3	P_4	Combined*
l_1 then l_2	1%	6%	94%	4%	94%
1_1 then l_3	0%	7%	0%	40%	0%
l_2 then l_1	95%	40%	97%	0%	96%
l_2 then l_3	5%	92%	98%	97%	95.6%
l_3 then l_1	3%	0%	20%	0%	0%
l_3 then l_2	96%	94%	95%	99%	96%

After the spatial probability calculation is performed for each pattern, a table can be created that gives all of the probabilities that events will occur at two given locations; as shown in Table 1. Table 1 shows the probability that pattern P_i occurs at location l_x then location l_y. For example, if P_1 is event A then event B, the first percentage in Table 1 means that for every 100 times event A occurs at l_1, event B occurs once during the correct time period at location l_2. The combined column shows the summation of the percentages in the rows that are greater than our minimum confidence value, 90%. A minimum confidence value is used so that a relationship between two elements is not weakened because it does not exhibit all of the potential patterns. The minimum confidence value chosen will depend on the strength of the relationships within the network. If the relationships are easy to detect, then a high confidence value can be used. However, this could cause some relationships to be incorrectly discarded; i.e. false negatives. If relationships are harder to detect a lower confidence value can be used. However, this can cause some incorrect relationships to be included; i.e. false positives.

From the data in Table 1 we can construct a generalized relationship graph; as shown in Figure 4. We can use this relationship graph to help us more accurately correlate future events. For example, if we receive events from R1 and R2, we would have a greater belief that they are related than if we received events from R1 and R3.

Figure 4. An example relationship graph

In addition to using the generalized relationship graph, we can also use the pattern specific spatial relationships to correlate events. This is useful as certain locations may be strongly linked by only certain patterns, and never linked by other patterns. Using a mixture of probabilities from the generalized relationship graph and a specific pattern we can correlate frequent patterns between locations while also allowing the possibility of new pattern occurrences between related locations.

5. EXPERIMENTAL EVALUATION

To test our method we used data from the core routers of a major telecoms company. As we only had a short period in which to collect data, we tested our method using only 1400 Syslog messages produced by 8 routers taken over a 22 day period. Figure 5 shows the physical topology of the network. The links between the routers are optical links, which in reality also pass through DWDM equipment. However, our analysis does not currently include the event logs for this equipment.

Our experiment consisted of giving a tool – which we created that implements our described method – a text file containing the Syslog data taken from the 8 routers. The tool then provides an output of how it thinks the routers are related. Our tool was not provided with the physical topology or any other information about the network; the links it discovered were from looking only at which locations logged events in patterns that it discovers.

Ideally, the number of event messages and the number of days over which the Syslog messages were collected would be greater. However, despite the small data set our initial results are promising. Figure 6 shows the relationship topology that our tool produces given the Syslog data; links that follow the actual topology are in black and other links in grey.

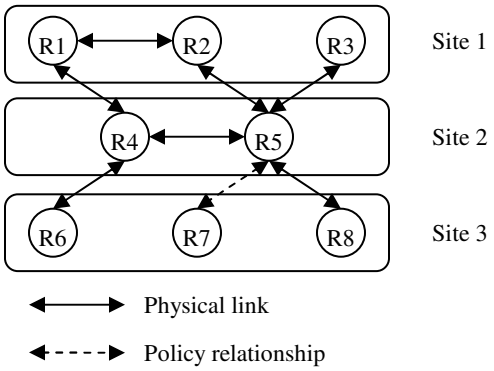

Figure 5. Core network topology[4]

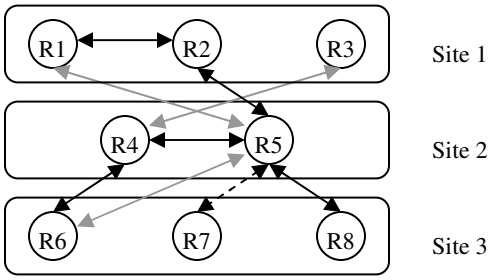

Figure 6. The computed relationship graph

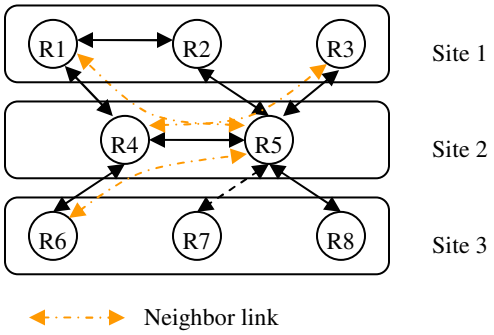

Figure 7: Categorized link types

As shown by Figure 6, the automatically generated relationship graph and the actual topology are similar. The automatically generated relationship graph links R1 to R5, instead of R4, R3 to R4, instead of R5 and has an extra link from R5 to R6. All of the additional links that were found are one degree of separation from the physical links of the network. These links can be explained by the neighbor relationship described in the Introduction; for example, R4 may exchange BGP information with both R5 and R6, and our spatial heuristic would see that R5 and R6 logged sequential events; these neighbor links are shown in Figure 7.

[4] In reality the core network has 13 routers; however, due to the small data set some routers did not produce enough events to be included in our analysis.

We think that our tool did not identify all of the physical links in the network due to the small amount of data used for the experiment, so not all of the links logged enough related events to produce a statistically significant relationship. However, the tool did identify the link between R5 and R7, which was the only non-physical relationship identified within our test network, showing that our method can discover more than just physical relationships.

Figure 7 shows the result if we superimpose the physical topology of the network over the automatically discovered relationships in the network. As shown, the grey links in Figure 6 can then be re-categorized as neighbor links because we then know that when there is a grey link connecting two routers they are physically connected to a common neighbor. We believe this further shows the ability to discover non-physical relationships. Were the grey links simply erroneous, we would also expect grey links from routers in Site 1 to Site 3 as there are 6 routers in these two locations combined, and only 2 routers in Site 2. However, this is not the case, none of the grey link incorrectly across sites.

6. ADDITIONAL GRAPH USES

In addition to helping us more accurately correlate events, the relationship graph can also help network operators view policies and dependencies in their network. As the relationship graph is created automatically from readily available data, the information it can provide comes at zero cost to the operator. To automatically classify the relationship types between network elements we can use the physical topology of the network. To find the physical topology of the network we can either use an automated method, such as analyzing configuration files, or allow an operator to enter their network topology manually.

The first step to discover the relationship types between two elements is categorizing hieratical links between elements, as given by the physical topology. The next step is to categorize other links given in the physical topology as physical links. Next, any links between elements that share a common neighbor are categorized as neighbor links. Finally, any links remaining are categorized as a policy/functional link; currently we cannot distinguish between policy and functional relationships.

As elements may have multiple links between them, for example, a physical link and a policy link, we can use the patterns that create each link type to learn what patterns create each link type. For example, if patterns 1 and 2 occur for links that we have categorized as policy links and a link we categorized as a physical link contains patterns 1, 2, 3 and 4, we would re-categorize the link to be both a physical link and a policy link.

From the detailed relationship graph a network operator can quickly see relationships that may otherwise take lots of time to discover. The detailed relationship graph can also quickly show an operator what other elements on the network may be affected if a change is applied to an element. For example, if a router is rebooted it may affect a router on the other side of the world that is physically 5 hops away if those two routers are linked by a policy. Also, as shown in Figure 7, some elements, such as R5, have many more relationships than other elements; this can help show operators which elements in their network are important to the network operation and which elements are not.

7. DISCUSSION

While the relationship graph can capture frequent pattern episodes between different network elements, it has the same disadvantage as any system using statistical frequency; it cannot capture infrequent patterns effectively. Our current method of dealing with infrequent events and patterns is to ignore them. We chose this approach as it fits with our original intent that frequent problems need to be fixed, where as a one off problem may not need much analysis. A possible improvement on ignoring infrequent events could be to cluster the events after an infrequent event, then when the infrequent event occurs again cluster the events after it again and take the union of the two clusters. As events have a severity level associated with them, we could assume that high severity events would create a larger cluster of events than a low severity event; however, more research has to be conducted to accurately find event clusters caused by infrequent events.

As networks are not static entities, we also think that it is important to add an aging function into our method. The aging function will discount the belief that two elements are related over time if there are no recent sequential patterns between the two elements. It will also discount the weight that a pattern contributes to the relationship believability if the pattern has not been seen on the network recently. This will help the relationship graph adapt to changes in the physical topology of the network and changes to policy and protocols.

In addition to looking for patterns of events in our data, we can also look for patterns of patterns using the same method. Figure 8 shows a series of patterns that contains a macro-pattern; A then B then C occurs at l_1 then l_2 two time intervals apart.

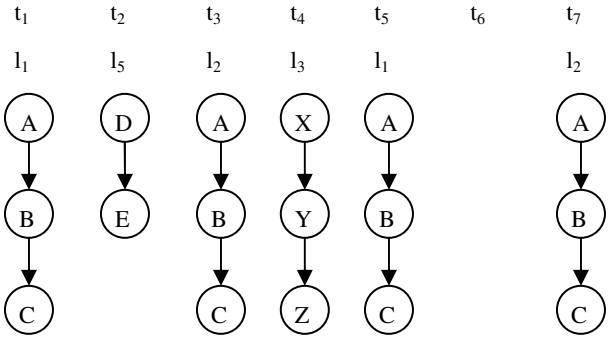

Figure 8. Macro patterns across locations

Finding macro-patterns across locations can help improve the accuracy of our relationship graph further, as a single event may not be the cause of an event at another location; it may be a series of events that causes another series of events.

8. CONCLUSION

The ever increasing size and complexity of networks has created the problem that no single person can know and control everything that happens on their network all of the time. Due to the importance of networks to business operations, it is also important that any faults in the network are quickly diagnosed and fixed.

We hope that by combining events from every network element under an operator's control, and applying our event correlation method, we can help operators more quickly analyze and solve faults on their networks. In addition to the relationship graph helping us more accurately correlate events, it could also show operators previously unknown relationships or dependencies in their network, and help show which areas of the network are most eventful – which could potentially mean they are less stable.

9. ACKNOWLEDGEMENTS

We would like to take this opportunity to thank Brian Cashman, Chris Small and Rick Summerhill, and Abilene in general, for providing us with data used in parts of our research.

10. REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Sept. 1994

[2] Cooper, G., Herskovits, E. A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning, Volume 9, Issue 4, Oct. 1992, Pages 309-347

[3] Srinivasan, A., Bhatia, D., Chakravarthy, S. Discovery of Interesting Episodes in Sequential Data, Symposium on Applied Computing 2006, Data mining, Pages 598-602

[4] B. Gruschke. A New Approach for Event Correlation based on Dependency Graphs, 5th Workshop of the OpenView University Association: OVUA'98, Rennes, France, April '98

[5] Valdes, A., Skinner, K. Probabilistic Alert Correlation, Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection 2001, Pages 54-68

[6] Sterritt, R. Discovering rules for fault management, Engineering of Computer Based Systems, April 2001. ECBS 2001, Pages 190-196

[7] Julisch, K., Dacier, M. Mining Intrusion detection alarms for Actionable Knowledge, SIGKDD 2002, Industry track papers, Pages 366-375

[8] Yemini, S., Kliger, S., Mozes, E., Yemini, Y., Ohsie, D. High Speed & Robust Event Correlation, Communications Magazine, IEEE, Vol. 35, Issue 5, Pages 82-90

[9] Sterritt, R., Marshall, A.H., Shapcott, C.M., McClean, S.I. Exploring Dynamic Bayesian Belief Networks for Intelligent Fault Management Systems, Systems, Man, and Cybernetics, IEEE Oct. 2000, Vol. 5, Pages 3646-3652

[10] Jiang, G., Cybenko, G. Temporal and Spatial Distributed Event Correlation for Network Security American Control Conference, June 2004, Vol. 2, Pages 996-1001

[11] Yamanishi , K., Maruyama , Y. Dynamic Syslog Mining for Network Failure Monitoring. Conference on Knowledge Discovery in Data, 2005, Pages 499-508

[12] Devitt, A., Duffin, J. and Moloney, R. Topographical Proximity for Mining Network Alarm Data, SIGCOMM'05 Workshop on Mining network data, Pages 179-184

[13] Bahl et al., Discovering Dependencies for Network Management., Proceedings of ACM HOTNETS-V, November 2006