MINING NETWORK LOGS: INFORMATION QUALITY CHALLENGES

(Practice Oriented)

James A. Pelletier
AT&T Labs - Research
jap@research.att.com

Tamraparni Dasu
AT&T Labs - Research
tamr@research.att.com

Abstract: Network logs are the key to many critical functions such as network security, network monitoring and network management. They play an important role in intrusion detection and early warning for potential worm and virus attacks. However, network log data are under-utilized -- largely ignored until the occurrence of an event that requires back tracking for diagnostic purposes. There are two main reasons why network logs are not subject to more rigorous analysis – the sheer volume and the inherent information quality challenges. In this paper, we use the context of classical data quality principles to outline some of the issues that we encountered, and the solutions that we devised, while working on a real network management application involving large amounts of network log data. While our discussion is centered on our case study, the problems we encounter and the solutions we devise are general and apply to a wide array of network log data and applications.

Key Words: Network security, network operations, data quality metrics, longitudinal analysis.

INTRODUCTION

Monitoring, maintaining and trouble shooting networks have always been a critical function. Recent events such as terrorist threats and cyber attacks on sensitive financial and medical data have added new urgency. However, device logs, which are the primary data that reflect network functioning, have occupied a low place in the data priority chain. They are mainly used as a diagnostic tool after an event of interest has already occurred. Quite often, they are discarded after a period of time, typically about a year or less. While enterprises take pride in keeping long histories of data for fickle customers, they seldom maintain the data that reflect the workings of the networks that enable them to serve their customers and run their businesses.

A common technique in network maintenance is to gather and scan logging messages from switches and routers for conditions that need correction. A large network can generate huge amounts of data, making the task difficult. Messages are often ignored, either because they are not immediately indicative of a serious condition, or because their infrequency is assumed to be a system anomaly. Occasionally, critical messages can be lost altogether in a mass of irrelevant data. Furthermore, the logs generated by these devices have limitations that create serious data quality issues. We believe that network device logs are an important data source that can yield valuable information for predicting network events but that the data

are often treated in a casual fashion. There are several reasons for this. Logging can generate large volumes of data and it can be difficult to pick a single performance-affecting event or trend out of all the normal messages being recorded. The "noise" level is very high. In addition, data quality problems make the data difficult to parse and understand. Extracting an intelligent assessment of what may be going on is not always simple or easy.

Classical statisticians make a distinction between "found" data and experimentally collected data. The latter tend to be planned, well thought out and carefully calibrated to meet analytical and experimental criteria. For example, in genetic and agricultural experiments, data are collected from plants that are carefully planted according to a well-defined experimental design [3]. In contrast, most modern data sets are found data – we have no control over how they are generated. Our ability to control and manage the data starts only at the data gathering stage. Network logs fall into the category of found data, characterized by a frustrating lack of consistency of formatting, semantics and a host of other data quality ailments.

Data Quality

The study and analysis of data has led to the definition of data quality principles. Classical criteria for "good" data include: consistency; uniqueness; timeliness; completeness and accuracy [7]. More recent work [2] has shown that in non-experimental settings such as enterprise operations, some of these criteria are neither measurable nor realistic. Updated definitions of data quality include operational criteria such as: successful end-to-end processing of data; extent of automation; interpretability of the data and measurability of data quality. Data quality dimensions are often inter-related, for instance consistency and automation, and it is difficult to choose a cut-and-dry data quality dimension to apply.

In this paper, we discuss the data quality challenges we encountered while building a tool for monitoring and predicting network events using network device log data. The tool itself is discussed in detail in a companion research paper [5] and merely provides context here. We give a brief outline of the tool; focus on the data quality challenges; and explain how they relate to classical data quality principles. For our application, we focus on process related data quality dimensions and metrics which often subsume classical metrics such as consistency of representation and uniqueness. Our main goal is automation of data processing to feed the network monitoring application. Therefore, we focus on the following metrics:

- Successful end-to-end processing
- Completeness
- Interpretability

Note that the learning from this case study is generally applicable to network log data and are not specific to this particular application. We found our experience with device log data to be completely transferable to network log data for a security application that we are working on. The issues of time stamps that are difficult to parse, irregular formats and a bewildering array of field separators are common to log data. Such data are becoming increasingly important in network monitoring and security, and are no longer a niche data type relegated to obscurity.

PROBLEM DEFINITION

We are interested in analyzing network device logs to identify historical patterns that are associated with network events of interest. For example, are potentially severe events preceded by distinctive patterns of less severe events? Or, do clusters of events observed on multiple devices share a common root cause? Automation of such diagnoses and associated remedies can make network maintenance more timely, robust and efficient.

The Application

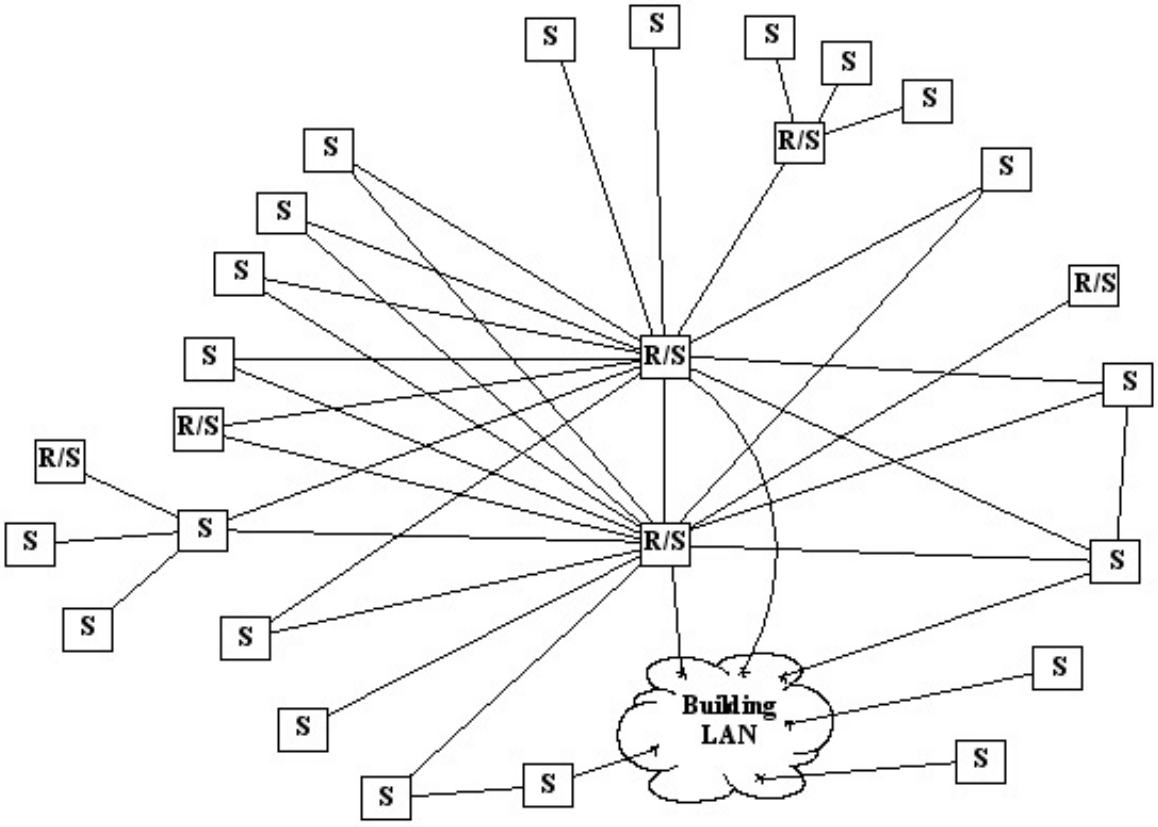
The tool is a mid-layer function in an agent architecture geared to monitoring large systems. Specifically, its aim is to examine logging data over a long interval for trends or activities that may not be readily apparent over a short period. One of the stated goals is to identify and automate useful techniques by folding them into an expert front-end. The expert system will have the ability to examine the historical data gathered, select the correct analytical method from a suite of methods, and extract some meaningful information about the state of the examined system. The suite includes statistical methods based on point processes and other discrete event models, in contrast to the approach based on clustering log messages in [9] and [10].

Existing tools are often limited to a pre-defined set of rules gathered from experts. A limitation of such an approach is that an event that is not in the rule base will not be flagged and the rules will only be as good as the domain expert who generated them [8]. We propose using statistical methods to discover associations and allow the expert system to pick the best statistical technique given the state of the network. Statistical analysis of complementary data sources such as netflows and packet level data can identify events that are not a part of the rule-base and hence result in adding new rules to the expert system.

The Data

We describe a general scenario below because we cannot reveal the details of the actual application for proprietary reasons. The data in question is log data gathered from the devices that form the bulk of an internal research network. The network, shown in Figure 1, consists of Cisco switches (S) built around a core router block composed of two Cisco Catalyst 6000 class switches with MSFC routing modules (R/S). HSRP (Hot Spare Router Protocol) coupled with network link redundancy between the distribution switches and the core block gives the network resilience and flexibility. Please see [6] for details.

We selected a small network comprised of hardware from a single vendor. The network was large enough to produce the type of data we needed and small enough to be manageable. Its hardware homogeneity ensured that we would avoid issues arising from a multiple vendor network. The expectation was that we could focus on our application without worrying about data issues but, as it turned out, this was not always the case.



.

FIGURE 1 – Network Topology

INFORMATION QUALITY CHALLENGES

In this section, we will discuss some commonly found data problems we encountered while monitoring the Cisco network and outline the solutions we developed. Our ultimate goal is to automate the treatment of glitches to derive a well-behaved dataset where each record conforms to some defined template. A consistent and clean data set permits further automation of analysis and prediction with minimal manual intervention. As experience shows, manual intervention often leads to the introduction of other human errors leading to further corruption of the data. A by-product of the automation of data cleaning is a general set of heuristics and rules for data quality assurance and data quality measurement that are widely applicable to log data.

Consistency

As mentioned above, consistency of format and representation is essential for automation, which in turn reduces cycle times and manual errors. Sometimes, the consistency issues were compounded by insufficient domain knowledge, context and expert opinions. We had to gather this additional information in order to resolve the consistency issues.

Record Consistency

A Cisco error message, as recorded by a Unix syslog, has four basic components arranged as

<Server Timestamp><Device><Device Timestamp><Error>

The details within each component are not necessarily consistent from one device to the other. The log data, as recorded by the syslog server, is considered "dirty" in the sense that there is no guarantee that two records picked at random will submit gracefully to the same parsing algorithm. Some of the disparities cannot always be controlled. In many cases, a quick resolution is reached with filtering scripts that use tools such as *grep, sed* and *awk*. Other examples are more complex and cannot be resolved without knowing the circumstances surrounding the records. Explaining that data with two different device names really came from the same physical device cannot be done without knowledge of each name's history. Logs simply do not capture this type of information; without this kind of knowledge, some records cannot be merged.

Device Names

Device names are a critical component in any analysis. Hence they must be correct and consistent across all records. Although our logging records stretched back almost 3 years, they were originally archived as a reflex rather than with any kind of long-term analysis in mind. When we began examining records, we discovered that some devices first appeared as IP addresses then later as DNS names. For any meaningful analysis, we needed to know that they were both the same device. For future records, changing device configurations to use the short DNS name as the standard was feasible. For existing records, correction was a nuisance but it could be done if you had access to the DNS. In some instances, this was not enough because we needed historical information that is not available from the DNS. We could not always state with certainty that a particular IP address had always been assigned to the same device. An exploratory analysis involving a simple plot of errors in time (Figures 2 and 3) revealed that two devices (red and green) occupied non-overlapping intervals of time but generated the same patterns of errors. Further investigation of external data showed that the two names referred to a device that had been renamed. This is an example of a case where an examination of the historical data revealed a curious fact that could be explained only through external data not stored in any log file or data source.

Error Message Variations

There is variability within the error message component of the log records. The logging data sent by a group of network devices may be consistent within a particular switch model and OS version, but there is no guarantee that this will be true for different device models or software versions. In addition, although there is overlap between IOS and CatOS messages, there are also some significant differences.

Software upgrades can introduce changes in the error messages being reported. In the following example, *ckore1* used facility **PAGP** to report that a port had left the bridge at 7:57. However, the same switch used facility **ETHC** to report the same action at 8:31. Both messages are reporting the same activity but the messages that did so are slightly different.

Apr 3 07:57:22 ckore1 2003 Apr 03 07:57:27 Eastern -05:00 %PAGP-5-PORTTOSTP:Port 4/1 left bridge port 4/1

Apr 3 08:32:20 ckore1 2003 Apr 03 08:31:08 Eastern -05:00 %ETHC-5-PORTTOSTP:Port 1/2 joined bridge port 1/2

The cause, in this case, was a CatOS upgrade. To mitigate this problem, we grouped messages according to the type of event being reported. For example, **PAGP-5-PORTTPSTP** and **ETHC-5-PORTTOSTP** were given the same group number because they were both reporting that a port dropped off the bridge. Group numbers were inserted into each record as a way of quickly finding a particular type of event without worrying about the different types of messages that could have generated it. We needed to devise this manual workaround in order to overcome the lack of standard representation. This is a common challenge in data mining activities, particularly those involving data federated from multiple sources or data from patchy legacy systems.

Date and Time Consistency

The timestamp recorded by the log server does not always agree with a device's timestamp. Normally, timing differences are less than a few seconds but improperly configured date and time functions can cause significant discrepancies. Enabling NTP (Network Time Protocol) on each device synchronizes them to the same master and reduces differences between the logging server and the network device [1].

For our dataset, using the date and time as recorded by the log server offered the only consistent reference point. Since not all devices were originally configured for NTP, this made event ordering an even more difficult problem [8] [1]. The delay between the time the event occurred and the time it was recorded by the server determined the amount of timing uncertainty. In some cases, we may never know for certain which event occurred first.

Interpretability

A major issue with found data is interpretability. The data becomes inaccessible to analysis when we cannot parse, understand and interpret the data accurately. An improper or incomplete understanding of the data can lead to incorrect analysis and misleading interpretation of the results. An obvious source of problems is lack of metadata such as data definitions and data dictionaries. Other problems include ambiguous representations and imprecise field separators.

Year Ambiguity

As shown below, not all devices include the year in their log messages.

(CatOS) *Aug 19 14:43:54 rdlb1 2004 Aug 19 14:32:22 EDT -04:00 %PAGP-5-PORTTOSTP:Port 2/46 joined bridge port 2/46*

(IOS) *Aug 3 14:03:47 rdlb4 213: Aug 3 17:53:42: %LINEPROTO-5-UPDOWN:Line protocol on Interface GigabitEthernet0/3, changed state to down*

In this example, *rdlb1*, a CatOS device, has supplied the year (2004) while, *rdlb4*, running IOS, has not. Ordinarily, this is not an issue because log records are not retained for any extended period. Consider the manner in which they are used – as a tripwire, an indication that a problem may exist. Immediacy is the goal, not long-term analysis.

Long-term retention is workable if the records are kept in the same order they were received and if some of them include the year. Since CatOS and IOS records are interleaved, a record containing the correct year can always be found. But inferring the year in this manner is risky at best. The difficulty arises when an IOS record is extracted from a dataset. Suppose a search seeks out all *rdlb4* errors. The resulting record set would contain only *rdlb4* records and any references or markers that might have indicated the year are lost. To compound the problem, manipulating or sorting the extracted records would remove any vague order that might have existed when the records were mined. There is no way to know if two records with the same month are even in the same year.

This issue is relatively easy to correct. The logging server archives the records by month with a file name of the form **<year><month>**, where the year is the full number (2004, not 04) and the month is its number (01 through 12). Since each month is preprocessed separately, the year indicated in file name is inserted in each record.

Field Separators

A particular concern is the variability in message formats from completely different sources [9]. Although the blank space is the most common field separator, other characters can also be used. Our dataset contained only Cisco devices so we only had to deal with Cisco's message format. Even so, log messages, as the following example shows, have more than one type of field separator.

May 1 09:53:31 klab1 2002 May 01 09:52:53 EDT -04:00 %SYS-2-PS_OK:Power supply 2 okay

The blank space is commonly used to delineate most of the fields. In the preceding example, the logging timestamp (May 1 09:53:31), device (klab1), device timestamp (2002 May 01 09:52:53 EDT -04:00) and error message (%SYS-2-PS_OK:Power supply 2 okay) are all separated by blank spaces. The fields within the two timestamp groups use both blank spaces and colons. The month, date and time use spaces while hour, minutes and seconds use colons.

The four-part error message uses both the hyphen and colon as delineators. Its structure is

%<Facility>-<Severity>-<Mnemonic>:<Description>

All the information between the percent sign and the colon defines a specific message. It contains the facility affected, the severity of the error and a mnemonic identifying the type of error. The description field after the colon gives details on the specific port, board, protocol, or service affected. To simplify matters, the facility, severity level and mnemonic are treated as a single field and the hyphens inside are ignored. The description following the first colon in the error message is also treated as a single field. This field can contain blank spaces, colons, percent signs and hyphens. Treating the description as a single field avoids the issue of using characters that can appear elsewhere in the log record as a field separator.

Hierarchical and nested representations require preprocessing to make the data accessible to standard analysis models. In our example, preprocessing replaces the original field separators with the pipe symbol (|). The pipe is a good choice because it does not appear within any of the fields.

Accuracy

Accuracy is perhaps the most emphasized data quality principle. However, it is difficult to enforce, especially in large data sets that are common in the data mining community. Text data presents its own challenges.

Data Validation

The log server stores the data as it is received without performing any validation tests. The server will accept any message as long as it passes syslog's general criteria – the message is of the right form and is sent to the right port and facility [4]. The contents of the message can be meaningless. Data verification is usually done by another application.

Despite using UDP, a connectionless unreliable protocol as the transport, garbled messages seemed to occur relatively infrequently. A few cases were discovered where characters were either inserted into or deleted from fields. Locating them can be a nuisance and the reason for the error is sometimes difficult to explain. In one instance, the three-character month field suddenly began appearing with four characters. For a four-hour period in 2003, logging messages from an MFSC module consistently reported the month as ".Jan" instead of "Jan". Messages from the CatOS side correctly reported the month as "Jan". After four hours, the aberration disappeared and was never seen again. Other devices with the same configuration never exhibited the problem. The best explanation is that it was a bug caused by an unusual set of circumstances. Correction, in this case, was simple, but tracking one-of-a-kind errors are an issue.

Completeness

Missing data are a commonly encountered problem in data mining and analysis applications. Individual observations, records, entire attributes, or even big sections of the data matrix can be missing. Gaps in temporal data are of concern because they might contain rare events (outliers) that could change the analysis and predictions significantly. The lack of log records can be the result of data being misdirected or lost rather than a smoothly running device. The absence of data does not necessarily imply the absence of nasty events. A device may have failed to recognize or record a significant or interesting event because

its logging level was set too low. Log messages may have failed to reach their destination because of a transmission error or because the log server was down. Whatever the reason, interesting data has been lost.

Since logs are event-driven, determining that there is a problem based on a lack of messages is difficult. Because of a tendency to ignore logging data until a problem occurs, the reason why events are missing may not be discovered until it is far too late. Once an event has occurred, references to it can only be found in log files. Any recovery months after an event has happened is usually impossible. Some inferences may be made from existing data but concluding that a system is in a particular state when data may be incomplete or missing is inherently risky.

Consistent Logging Levels

A hidden form of missing data is non-uniform levels of data from different sources. Low or inconsistent logging levels affect the quantity of data being delivered. While the presence of data is more important than its absence, the more data you have, the more options you have. Mining data from multiple devices can be more difficult when not all of them are supplying the same level of detail.

The severity level is a single digit that describes how serious an event is. The range is from 0 (system unusable) to 7 (debugging). As you would expect, 0 produces few messages while 7 is a veritable flood of data. The best logging level is the one that produces enough data without interfering with the performance of the device. A reasonable choice is 5 or 6. The biggest difference between the two is that level 6 generates data on actions that may be contextually significant. Reporting that a configuration block has been changed and identifying the culprit can be helpful when trying to explain a sudden burst or absence of errors.

Domain Knowledge

Knowledge of the domain and features peculiar to it are an important component in the correct understanding, use and interpretation of the data. For instance, peculiar data representations to accommodate the specific needs of an application are quite common. These conventions tend to be undocumented and are often lost when the experts leave. We describe below an instance where context plays an important role in interpreting and solving the data quality issues.

The Role of Context

While examining a trend in one of our devices, a curious absence of data occurred in two time periods. One of them was explainable, the logging server had been disabled during one of the intervals, but the second one was a puzzle. What had happened? Why was data available for other devices but not this one? Was the lack of data significant? Because the events had occurred over two years earlier, no one could remember what could have caused the gap. This situation illustrates the significance of maintaining a history of external events. Part of the power of log files resides in event sequencing. To some extent, the implication of those events depends on other actions that may have occurred at the same time but were not recorded within the log file.

One of the biggest operational failures is that log files are only examined when something bad happens. During troubleshooting, correlating a possible external action that could have led to a logged event is painless. A configuration change that was performed yesterday can be easily remembered; a name change that happened two years ago is more difficult. In one analysis, the relocation and renaming of a switch (Figures 2 and 3) led to some confusing results. If a history of the physical device had existed, we would have known immediately that the two different sets of records could be combined. We were able to sort out the “context” by examining a number of external resources - old emails, one brief entry in a logbook, and an archived document that included a list of switch names, their IP addresses and the networks they supported. This case is typical. Organizations do not normally systematically collect and retain this kind of information because of the administrative overhead.

Information about external events is useful for two reasons. First, there were too many things we couldn't easily explain without knowing the circumstances surrounding the events. Aberrations could have been explained if knowledge about name changes, IP changes, software upgrades, device moves and policy changes had existed. The disappearance of a particular message and the appearance of another could only be explained after we had assumed that an upgrade might have taken place, then backtracking the upgrade notes until we found the right one. Second, separating or merging records without knowing a device's pedigree, history or its operational environment can have unexpected results.

Contextual information relevant to logged events relies primarily on administrator recollections. We've never met an admin with an eidetic memory so it should not be surprising if events that occurred some time ago should be fuzzy. Recalling a set of circumstances that occurred a few weeks ago is feasible; remembering the exact conditions after a few months is not. To some extent, increasing the severity level is useful but what is really needed is a method to collect the actions taken or the events observed as timestamped log records. By inserting records of external events into the logs, context can be integrated into the temporal flow. Because external context-specific events happen far less frequently than normal logging events, the number of messages generated is small and should not pose a burden to the log server. The difficulty is not the mechanics of inserting records in log files; the hard part is the discipline required to do it in an organized and timely fashion.

EXPLORATORY ANALYSIS

We present below a small example of exploratory analysis of historical network device logs. Details of the actual network-monitoring tool built using longitudinal models and an expert system front-end are found in [5].

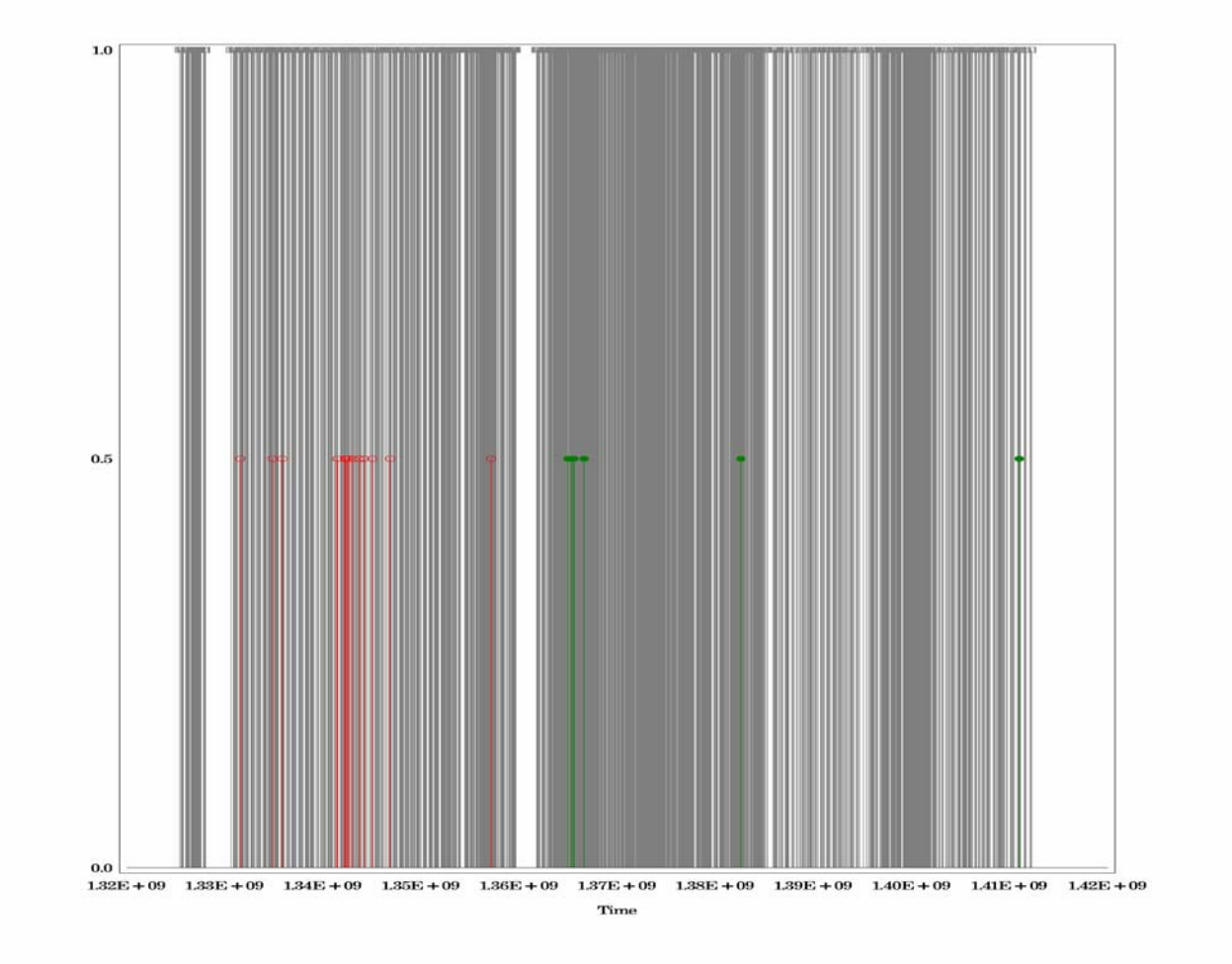


FIGURE 2 – Error distribution across time

Figure 2 shows a minimalist representation of errors of different kinds observed across 5 devices over a 2-3 year time period. We have suppressed titles, axes and legend for proprietary reasons, without losing the general information. The X-axis represents time (granularity=seconds) and the Y-axis identifies the device, re-enforced by color. In order to make a point, we have isolated two devices (red and green) and shown all the others in gray. The presence of a needle indicates an error on that device. There are two interesting observations: (1) The two gaps in the data, the first caused by an intentional turning off of the logging feature and the second gap (half way through the time period) caused by unknown factors. The gaps in logs are observed across all devices. (2) The renaming of a device half way through the observational period (initially represented in red with a circle plotting symbol and later in green with a dot for a plotting symbol.) The renaming of the devices is quite evident here where the devices are present in clearly non-overlapping intervals. A tool that relied on pre-defined "rules" for detecting network problems would not have identified this kind of an error. For precisely this reason, we need exploratory analysis that can bring to light errors that are not previously known or not included in the set of rules.

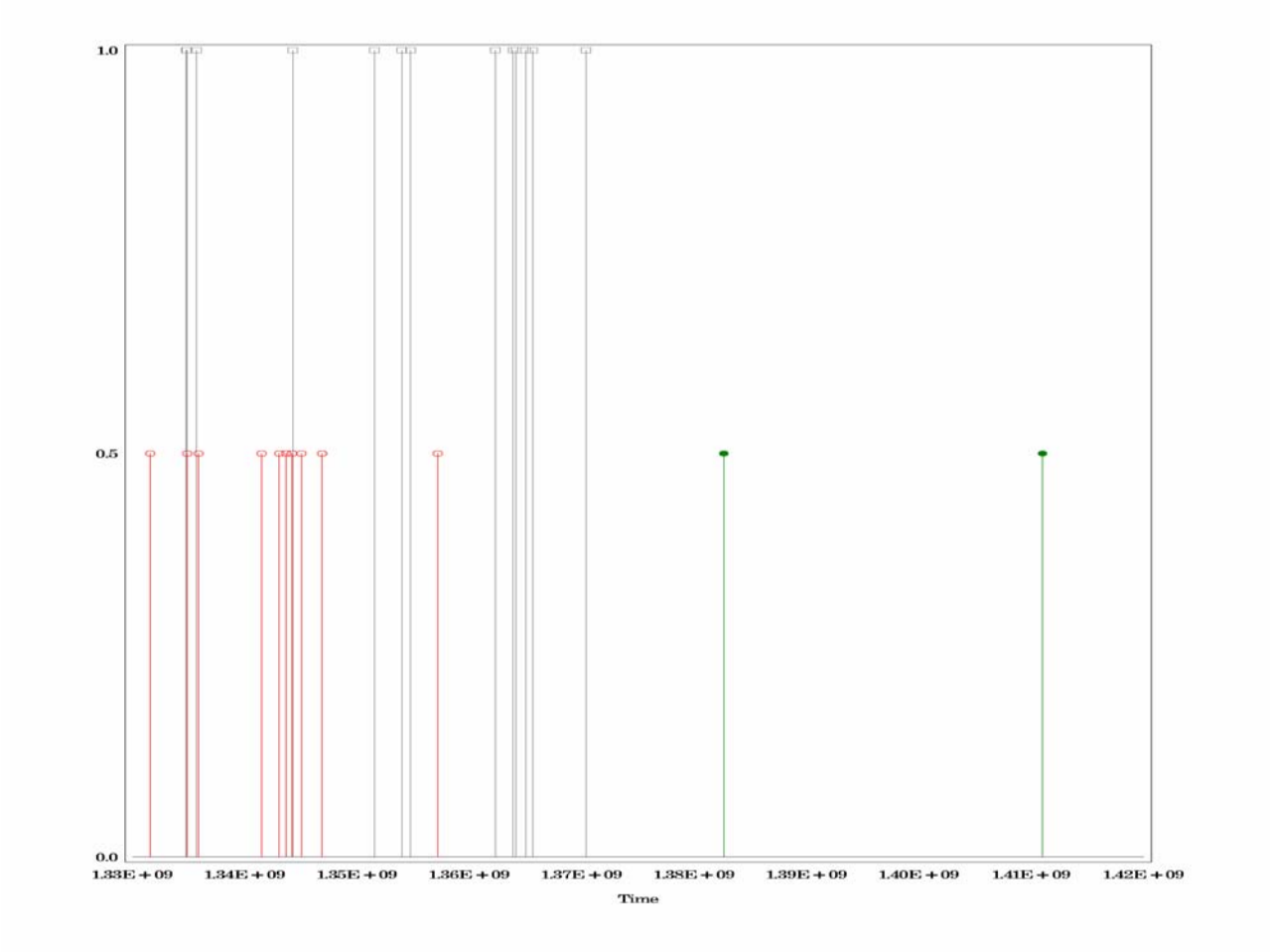


FIGURE 3 – "Errors" caused by scheduled events

In Figure 3, we show a particular kind of error related to a power supply. This is usually a high severity error but is often caused by anticipated maintenance that is scheduled well in advance. The regularity in the errors is an indication of this. Clusters of power supply errors are often a side effect to troubleshooting. Here too, we notice the renaming of devices evident in the error patterns. Sometimes, the absence of an 'event' is significant – for instance, the unusually long gap between the two green needles could indicate the failure of a scheduled maintenance event. Similarly, the absence of the grey needles beyond the half way time point might indicate a change in devices or logging levels that must be recorded in the context database for future analysts. There are many other interesting events that we do not include here due to space constraints.

DATA QUALITY METRICS

While there are many data quality metrics that we could discuss, due to space constraints we mention just a few that are of special interest from a preprocessing perspective.

- **Successful end-to-end processing:** When we started our analysis, a good 30% to 40% of the data was rejected by the data mining algorithm. Because of the non-standard representation of data, records could not be parsed properly, leading to many values being read as missing or invalid. However, after we applied our preprocessing filter to tease out the fields using a pipe separator and enforced a standard format for a given field, the data mining algorithm accepted 99.999% of the data. So the percentage of data accepted by the analysis algorithm serves as a crude but effective preliminary data quality metric.

- **Completeness:** As shown in Figure 2, there was an unscheduled lapse in logging that accounted for approximately 2% of the data that was logged during the analysis period, depending on how you estimate the number of lost records. While we could not recover this data, we have established procedures for preventing this in the future. As a part of this procedure, we have instituted "context databases" that help with the next metric as well. Note that despite our best efforts, there will always be missing, incomplete or damaged data. The best one can hope for is to be aware of such data and make statistical adjustments in the analysis to account for biases introduced by such situations. Statistical techniques such as missing value imputation and ignorable and non-ignorable patterns of missing data can help in making such adjustments.
- **Interpretability:** Inadequate information can lead to misinterpretation of the data and incorrect conclusions. For instance, our example of a device name change would have given completely different results if we hadn't recognized that two different names referred to the same device. In this case, we drew on a body of external data that put the anomaly in perspective. By contrast, we were able to explain a cluster of port security violations caused by a summer visitor trying to plug his laptop into a forbidden network because we realized what the sequence of log messages meant. In both cases, our knowledge was broad enough to recognize the problems and devise ways of resolving them. We were able to identify and explain almost 100% of the anomalies we found during the exploratory analysis because we successfully linked log events to external data sources. The implication is that an expert system tasked with data interpretation would require an extensive and flexible body of domain knowledge as well as access to information sources outside the device logs. It must recognize when the log messages by themselves are sufficient and when additional information is necessary.

CONCLUSION

Network device logs are an important data source for monitoring and predicting network events and should not be ignored because of its daunting size or inherent messiness. We have presented an overview of data challenges commonly encountered when dealing with network device logs. These range from inconsistent representations that make automation difficult to interpretation issues that require knowledge outside the data and metadata. Our experience shows that preprocessing is almost always necessary if a found log set is to be translated to a more manageable form. There are a number of points that should be kept in mind if long-term analysis is the goal. We summarize our experience below:

Configure everything with long-term analysis in mind: Log files are commonly used as a short-term troubleshooting tool without regard to what the long-term implications might be. Set device configuration standards for things like naming, and timing (NTP) and apply them consistently.

More data is better: Set logging to the highest level that doesn't interfere with the operation of the device. Filtering data is easier than working around missing or incomplete data.

Keep everything: There is a tendency to delete log files after too short a time. Device failures to include the year in their timestamps can be corrected if historical data are available. If storage is a problem, use off-line storage.

You need context: Knowing about external events is a valuable asset. Keep track of anything that might have a bearing on the logs either by inserting timestamped messages in the log file or by maintaining a separate database.

Normalize the data: Even with our Cisco logs, there were variations between devices. Adding different devices makes the problem worse. Preprocess log files to conform to a standard template.

Examine the log files: Mine the data regularly using any techniques or tricks that might isolate unusual or interesting patterns.

While our advice is a specific interpretation of the data quality principles in the context of network device log data, it is also applicable to other types of data. A parallel effort investigating IDS (Intrusion Detection System) and network vulnerability logs has revealed similar problems to those we faced while evaluating network device logs. Our experience with network device logs and the techniques we used can be applied directly to these logs and, by extension, to other types of log data.

Future Research

Historical network logs can be "mined" to discover predictive patterns to anticipate and perhaps prevent undesirable network events. This is in contrast to monitoring the network logs for pre-specified rules. We employ longitudinal models based on point processes and other discrete event models to identify predictive patterns. We combine this with an expert system front-end to automatically select the most appropriate analytical model given the recent log history. Details can be found in [5], under preparation.

REFERENCES

[1] Bing, M. and Erickson, C. Extending UNIX Logging with SHARP, *Proceedings of the 14th Systems Administration Conference*, New Orleans, LA, Dec 3-8, 2000.

[2] Dasu, T. and Johnson, T. *Exploratory Data Mining and Data Cleaning.* John Wiley & Sons, New York, NY, 2003.

[3] Fisher, R. A. *The Design of Experiments*. Oliver & Boyd, Edinburgh, 1935.

[4] Lonvick, C. The BSD syslog Protocol, RFC3164 (2001).

[5] Pelletier, J., Dasu, T. and Vesonder, G. T. Expert Systems that use Point Processes for Network Management. In preparation, 2005.

[6] Quinn-Andry, T. and Haller, K. *Designing Campus Networks*, (Cisco Press) Macmillan Technical Publishing, 1998.

[7] Redman, T. *Data Quality: Management and Technology.* Bantam Books, New York, NY, 1992.

[8] Rouillard, J. P. Real-time log file analysis using the Simple Event Correlator (SEC), *Proceedings of the 18th Large Installation System Administration Conference*, Atlanta, GA, Nov 13-19, 2004.

[9] Vaarandi, R. A Data Clustering Algorithm for Mining Patterns From Event Logs. *Proceedings of the 2003 Workshop on IP Operations and Management*, pp119-126, 2003.

[10] Vaarandi, R. A Breadth-first Algorithm for Mining Frequent Patterns from Event Logs, *Proceedings of the 2004 IPIP International Conference on Intelligence in Communications Systems*, LNCS Vol. 3283, pp 293-308, 2004.