# FALCON: Fault Management via Alarm Warehousing and Mining

Matt Grossglauser
AT&T Labs-Research
mgross@research.att.com

Nick Koudas
AT&T Labs-Research
koudas@research.att.com

Yongseok Park
Coree Networks
yongseok@coreenetworks.com

Alice Variot
AT&T Labs-Research
variot@research.att.com

**Abstract**

The ability to manage faults in large scale networks is of vast importance for successful and effective network management operations. In this paper, we describe FALCON, a project underway at AT&T Labs-Research focusing on fault management in IP networks. FALCON's goal is to automate various fault management tasks through warehousing and mining technologies.

We describe FALCON's architecture and comment on its components. We present ALVIS, an alarm navigation and visualization tool developed to explore alarms generated by AT&T's IP backbone. ALVIS displays alarms of multiple network elements as a set of state time series in a single picture, thereby revealing interesting structure and anomalies in the data. We also describe KALHAS, FALCON's data mining agent, currently under development, and outline our vision and current research agenda towards fault management automation through successful development and deployment of state of the art incremental mining and stream computation techniques.

## 1 Introduction

IP networks are growing in several dimensions, including the number of nodes and routers, the types of applications and services supported by the network, the number of customers and the classes of traffic. This increase in scale goes hand in hand with an increase in complexity, because it fundamentally complicates distributed control mechanisms, such as resource allocation and routing. Since the number of network elements increases, the probability of faults occurring somewhere in the network increases as well. Faults occur more frequently, but they also propagate more widely, since services and resources rely on each other. For example, if a router fails, the network will be partitioned; the management system will then generate alarms for every network element outside its partition, because these elements cannot be reached by polling packets any longer.

The increase in scale and complexity hampers efficient network management operations, which in turn incurs costs and affects overall network reliability. Faults occur so frequently and trigger such a large number of alarms that network operators quickly become the bottleneck in the recovery process. The alarm information is too low level and too redundant for operators to interpret in real-time.

Fault and ALarm COrrelation in IP Networks (FALCON) is a project underway at AT&T Labs research, aiming to automate much of the fault management process through deployment of advanced warehousing, mining and visualization techniques. In this paper we describe FALCON's architecture, and we elaborate on various architectural components. This paper is organized as follows: Section 2 introduces the problem of fault management and the challenge of developing robust alarm correlation rules. Section 3 describes FALCON's system architecture and elaborates on ALVIS, FALCON's visualization client and KALHAS, FALCON's data mining module currently under development. Section 4 concludes the paper and outlines our current research and development efforts.

## 2 Complexities of the Fault Management Problem

In a large, heterogeneous communication network whose topology and configuration is subject to frequent changes, the occurrence of physical and logical faults is unavoidable. The goal of fault management is to avoid faults or limit their effect. Managing a fault involves three steps: detection, diagnosis (i.e., finding the root cause), and recovery.

Detecting and diagnosing a fault is challenging in general because (a) a single fault can give rise to a multitude of symptoms, as mentioned above, and (b) each symptom viewed in isolation may be explained by a wide range of faults. It is difficult for a human operator to extract the correct root cause from all outstanding alarms in real time. This results in operator inefficiency and slow recovery from failures.

*Alarm correlation* [JW93, YKM+96] raises the semantic level of alarm information presented to human network operators and eliminates redundancy in the alarm stream as far as possible. An ideal alarm correlation system would uniquely identify any fault (or set of faults) in the network unambiguously and in real time. This ideal has remained elusive, for several reasons:

- The number of possible faults is extremely large.
- The *system model* that describes how a fault results in a set of symptoms is difficult to describe accurately and exhaustively. This system model depends on the physical properties of network elements, the software running in different components of the network, interdependencies among subsystems, etc.
- The management system itself can introduce noise and delay into the alarm stream. For example, an alarm storm caused by a severe fault can overload the management system, leading to dropped and delayed alarms. Also, the fact that in IP networks, management information is usually carried in-band (i.e., by the managed network itself) is problematic, because some faults (such as a network partition) may actually result in the suppression of some alarms.

Current alarm correlation systems therefore limit themselves to removing some basic redundancy from the alarm stream. This is typically achieved through a rule-based system. The rule base encodes known, straightforward dependencies in the system (e.g., if a router fails, ignore all failed polls to links terminating at this router). This kind of rule can be developed directly from expert knowledge about the system.

To reap additional benefits from alarm correlation, rules have to be developed that detect more intricate fault scenarios and eliminate more redundant symptoms. The development of such complex rules that are robust despite the noise and uncertainty in the alarm stream requires a more systematic rule development process. In particular, given that it is difficult to know and accurately characterize a system model (even for individual components, such as routers, links, firewalls etc.), we cannot rely on expert knowledge alone.

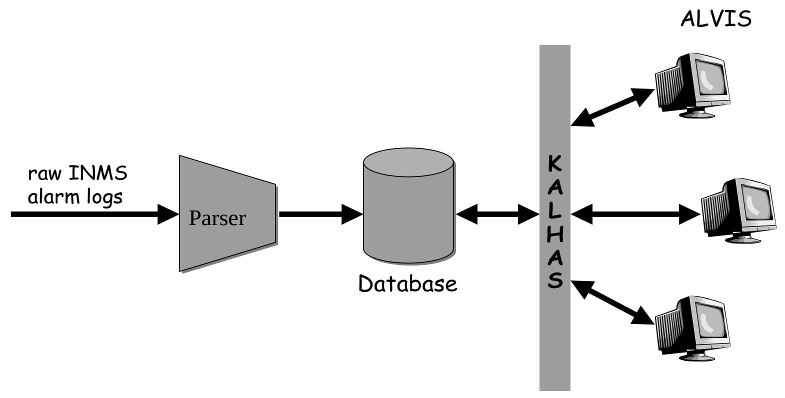


Figure 1: FALCON architecture

Rather, historical alarm data should be exploited to develop an accurate system model (e.g., by finding additional symptoms related to a failure) and to simulate and test candidate rules. In the FALCON project, we develop such an alarm warehousing environment where we combine visualization and data mining technology to assist in the rule development process.

## 3 System Architecture

Figure 1 presents the overall system architecture. We periodically collect alarm log files generated by the HP OpenView management platform. The alarm logs are in a human-readable text format; this format depends on the alarm type. In its raw form, the alarm logs are not amenable to efficient processing and querying, and they are very space-inefficient. Consequently, raw alarm files are parsed, generating relational tables suitable for subsequent processing using object relational technology.

The parser is a crucial component since it determines the granularity of the information transfered into the database. Enough information is extracted so that an informative temporal view of the data is recorded in the database. The form of the temporal information extracted varies depending on the type of alarm and network element. In its simplest form, for each network element present in the raw alarm log file, a state time series is constructed, recording the reported state of the network element at each time step, for the time interval corresponding to the raw alarm log processed. In a degenerate case the time series can be binary (the network element is reported active or inactive at each time step). In their general form, state time series can have multiple states.

All the information collected by the parser is loaded into an alarm warehouse. Information is appended into tables, either inserting information about new network elements or inserting network element state information about the current time interval. Since each entity in the database is tagged with a time-stamp, temporal management is natural. One, for example, can choose to always maintain in the warehouse data corresponding to a time window of interest (say 1 year worth of alarm information). The warehouse has been organized in a star schema for each type of alarm. Natural hierarchies exist in the data and can be utilized at the time of design. Various inclusion relationships exist between network elements; for example several interfaces in a router belong in the same interface card. Several interface cards belong to the same router etc. Such relationships are also of interest as their existence provides various schema design choices.

Querying the warehouse is performed through FALCON's ALarm VISualization (ALVIS) client, which we describe next.

## 3.1 Alarm Visualization

ALVIS is an interactive front end to the alarm warehouse; it is written in Java using the Swing API and interfaces to the object relational data store using JDBC. Using ALVIS one can take full advantage of the natural hierarchies existing in the data; ALVIS allows OLAP style of querying and data exploration, traversing hierarchies and interactively examining state information at every node. Queries are formed dynamically as the hierarchy is traversed. Moreover the user can dynamically choose to set ways to associate (essentially form join predicates) inter or intra hierarchy levels. Moreover, the visualization client places no restriction between the association of various database entities, thus the user can make use of knowledge about data semantics and associate elements in meaningful ways.

One of our design goals was to disassociate the visualization client from the semantics of the underlying data. We choose to do so for extensibility, i.e., so that new alarm types can be introduced without modification to the ALVIS code base. To visualize alarms, we model the alarm data as a set of abstract objects, which may represent network elements, such as routers, links etc., or logical objects, such as BGP sessions. Each object has a single state time-series associated with it. Then, modeling the alarm data involves three steps: (a) identifying what alarm types are of interest; (b) identification of objects within an alarm type (e.g., a unique link object might be identified by the name of its router and its local name on that router; a BGP session might be identified by the IP addresses of both participating routers); (c) definition of the states the object can be in (e.g., up-down for a link object, discretized utilization for a link load object, the states of the finite-state machine for a BGP session object, etc.).

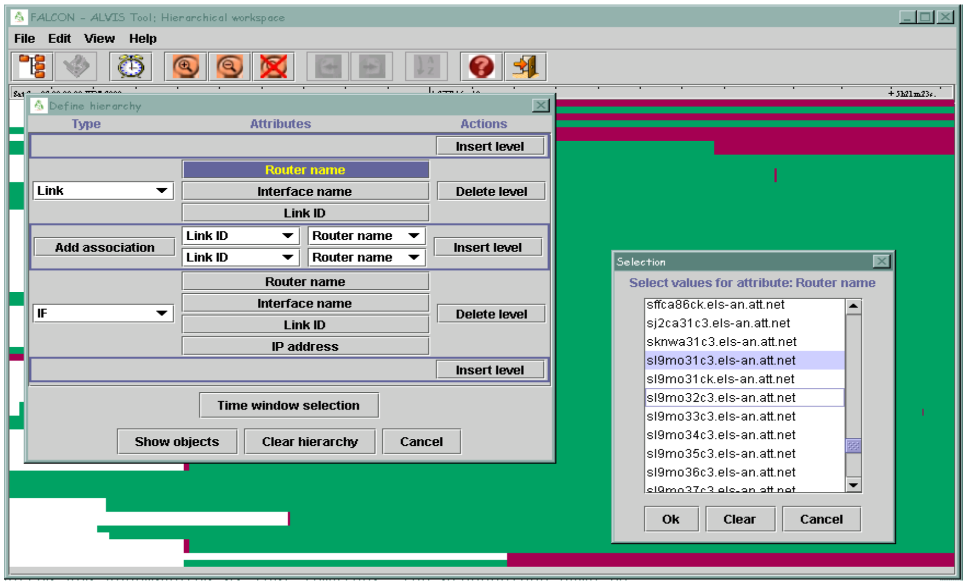
Addition of new alarm types can take place without interfering with ALVIS code. Only the parser has to be extended, and some meta-data describing the new alarm has to be added to the alarm warehouse. ALVIS relies on this meta-data to render the new alarm type as desired. The meta-data are populated the first time a new alarm type is introduced and are recorded by the parser on suitable meta-data tables.

Query results are presented in a graphics area displaying state information for each network element in the query result, for the temporal window selected. Query results can be reordered dynamically to aid visualization and visual exploration of the network element's state. Figure 2 presents a typical interaction with the visualization client. The user chooses to associate information about two network elements (interfaces and links). The domain of possible values (extracted from the database) is dynamically presented and the suitable network entities can be selected and folded into query expressions which are dynamically generated. Figure 3 presents a typical view of query results offered by ALVIS. The associated network elements are displayed, along with other attributes of interest and the corresponding state time series. Correlations between the state of various network entities can be visually revealed; consider for example Figure 3. A set of interfaces is reported as inactive approximately the same time and some of them stay inactive for approximately the same period of time.

## 3.2 KALHAS: FALCON's Data Mining Agent

The ultimate goal for effective fault management is to ease the network operator's task. Ideally, the network operator should be presented with the most likely faults causing the observed symptoms, along with suggested corrective actions. Traditionally, rule based alarm correlation was used to raise the semantic level of the information presented to operators and to eliminate redundancy. Such an approach has two main drawbacks. First, it is difficult to come up with the right set of rules, and second, it is hard to assess the impact of a new rule when it is added to a large existing rule based system.

The incremental nature of data collection presents the opportunity to identify and isolate the real network



Link: Router name, Interface name, Link ID

Link ID – Router name
Link ID – Router name

IF: Router name, Interface name, Link ID, IP address

Select values for attribute: Router name

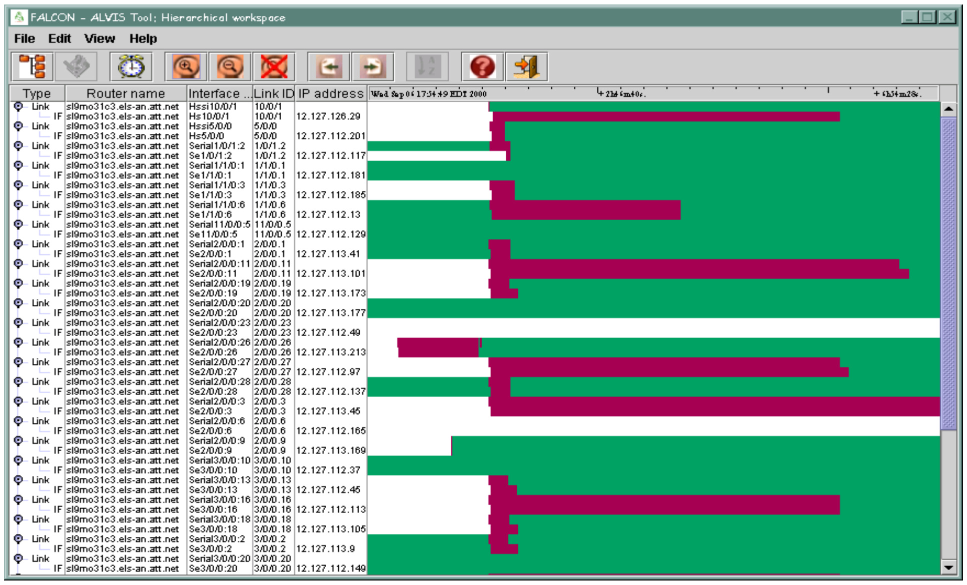
sffca86ck.els-an.att.net
sj2ca31c3.els-an.att.net
sknwa31c3.els-an.att.net
sl9mo31c3.els-an.att.net
sl9mo31ck.els-an.att.net
sl9mo32c3.els-an.att.net
sl9mo33c3.els-an.att.net
sl9mo34c3.els-an.att.net
sl9mo35c3.els-an.att.net
sl9mo36c3.els-an.att.net
sl9mo37c3.els-an.att.net

Figure 2: Forming Queries

problem online as the data arrive. Our vision is to assess alarm correlation online and examine the extent to which one can dispense the use of rules for this problem. This poses interesting incremental mining questions which in turn lead to challenging data stream computation issues. Consider the following scenario: A network operator instructs KALHAS to monitor a specific interface; KALHAS automatically initializes structures and employs algorithms to incrementally correlate multiple pieces of information available for this interface; when KALHAS deduces that the interface is no longer active it notifies the operator. When higher level entities (e.g., network cards) are monitored, the state of lower level entities is automatically and incrementally maintained to assess the state of the higher level entity.

To address such issues, we are designing an incremental data mining module. The mining layer lies between the database and ALVIS and interacts with both components. The operator interacts with KALHAS through ALVIS, selectively setting network components for subsequent monitoring. For each selected network element, KALHAS maintains state information incrementally, using its own address space and data structures, as new data arrive and updates its structures to reflect and identify changes in the data that lead to potential problems in the monitored entity. If the state information for each monitored element was accurate, identifying problems would be an easy task. However, state information is erroneous, thus multiple sources of state information (polled and pushed) should be correlated in an online manner to collectively reach a consensus about the current state of the network element. Designing such online techniques to assess correlation is an interesting problem. Commonly, network elements encompass sets of of other network elements (a set of interfaces in a network card, a set of cards on a router etc) which are also candidates for monitoring; effective techniques are required to asses the state of the higher level entity (e.g., a network card) by the assessed state of lower level entities (its network interfaces), in an online manner. Monitoring utilization of various network elements can also provide valuable information about network faults. Such monitoring can be effectively performed and visualized using incremental clustering techniques [GMMO00, GKS01].

Since we envision applications that involve network operators that take actions when problems occur, the quality of mining performed to infer the state of network elements on demand can be directly correlated with the actual state when the problem is resolved. This provides a unique opportunity to acquire online feedback


FALCON - ALVIS Tool: Hierarchical workspace
File
Edit
View
Help
Wed Sep 06 17:34:49 EDT 2000
+2h 6m40s.
+6h34m28s.

| Type | Router name | Interface ... | Link ID | IP address |
|---|---|---|---|---|
| Link | sl9mo31c3.els-an.att.net | Hssi10/0/1 | 10/0/1 | |
| IF | sl9mo31c3.els-an.att.net | Hs10/0/1 | 10/0/1 | 12.127.126.29 |
| Link | sl9mo31c3.els-an.att.net | Hssi5/0/0 | 5/0/0 | |
| IF | sl9mo31c3.els-an.att.net | Hs5/0/0 | 5/0/0 | 12.127.112.201 |
| Link | sl9mo31c3.els-an.att.net | Serial1/0/1:2 | 1/0/1.2 | |
| IF | sl9mo31c3.els-an.att.net | Se1/0/1:2 | 1/0/1.2 | 12.127.112.117 |
| Link | sl9mo31c3.els-an.att.net | Serial1/1/0:1 | 1/1/0.1 | |
| IF | sl9mo31c3.els-an.att.net | Se1/1/0:1 | 1/1/0.1 | 12.127.112.181 |
| Link | sl9mo31c3.els-an.att.net | Serial1/1/0:3 | 1/1/0.3 | |
| IF | sl9mo31c3.els-an.att.net | Se1/1/0:3 | 1/1/0.3 | 12.127.112.185 |
| Link | sl9mo31c3.els-an.att.net | Serial1/1/0:6 | 1/1/0.6 | |
| IF | sl9mo31c3.els-an.att.net | Se1/1/0:6 | 1/1/0.6 | 12.127.112.13 |
| Link | sl9mo31c3.els-an.att.net | Serial11/0/0:5 | 11/0/0.5 | |
| IF | sl9mo31c3.els-an.att.net | Se11/0/0:5 | 11/0/0.5 | 12.127.112.129 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:1 | 2/0/0.1 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:1 | 2/0/0.1 | 12.127.113.41 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:11 | 2/0/0.11 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:11 | 2/0/0.11 | 12.127.113.101 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:19 | 2/0/0.19 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:19 | 2/0/0.19 | 12.127.113.173 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:20 | 2/0/0.20 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:20 | 2/0/0.20 | 12.127.113.177 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:23 | 2/0/0.23 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:23 | 2/0/0.23 | 12.127.112.49 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:26 | 2/0/0.26 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:26 | 2/0/0.26 | 12.127.113.213 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:27 | 2/0/0.27 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:27 | 2/0/0.27 | 12.127.112.97 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:28 | 2/0/0.28 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:28 | 2/0/0.28 | 12.127.112.137 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:3 | 2/0/0.3 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:3 | 2/0/0.3 | 12.127.113.45 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:6 | 2/0/0.6 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:6 | 2/0/0.6 | 12.127.112.165 |
| Link | sl9mo31c3.els-an.att.net | Serial2/0/0:9 | 2/0/0.9 | |
| IF | sl9mo31c3.els-an.att.net | Se2/0/0:9 | 2/0/0.9 | 12.127.113.169 |
| Link | sl9mo31c3.els-an.att.net | Serial3/0/0:10 | 3/0/0.10 | |
| IF | sl9mo31c3.els-an.att.net | Se3/0/0:10 | 3/0/0.10 | 12.127.112.37 |
| Link | sl9mo31c3.els-an.att.net | Serial3/0/0:13 | 3/0/0.13 | |
| IF | sl9mo31c3.els-an.att.net | Se3/0/0:13 | 3/0/0.13 | 12.127.112.45 |
| Link | sl9mo31c3.els-an.att.net | Serial3/0/0:16 | 3/0/0.16 | |
| IF | sl9mo31c3.els-an.att.net | Se3/0/0:16 | 3/0/0.16 | 12.127.112.113 |
| Link | sl9mo31c3.els-an.att.net | Serial3/0/0:18 | 3/0/0.18 | |
| IF | sl9mo31c3.els-an.att.net | Se3/0/0:18 | 3/0/0.18 | 12.127.113.105 |
| Link | sl9mo31c3.els-an.att.net | Serial3/0/0:2 | 3/0/0.2 | |
| IF | sl9mo31c3.els-an.att.net | Se3/0/0:2 | 3/0/0.2 | 12.127.113.9 |
| Link | sl9mo31c3.els-an.att.net | Serial3/0/0:20 | 3/0/0.20 | |
| IF | sl9mo31c3.els-an.att.net | Se3/0/0:20 | 3/0/0.20 | 12.127.112.149 |

Figure 3: Viewing Query Results

about mining quality that could potentially be useful to subsequent mining operations. This calls for the design of incremental algorithms capable of incorporating such feedback in an online fashion. Moreover, actions taken by operators to resolve problems identified, can indeed be mined to derive suggestions about possible course of action when a related problem arises.

## 4 Conclusions

Effective fault management has the potential to improve network reliability and operational efficiency. We have outlined our current and future work in the context of FALCON, a project underway at AT&T Labs-Research to address fault management issues. We believe that this problem provides an excellent opportunity for interdisciplinary research, utilizing state of the art techniques from networks, databases and data mining technologies.

## References

[GKS01] S. Guha, N. Koudas, and K. Shim. Data Streams and Histograms. *Symposium on the Theory of Computing (STOC)*, July 2001.

[GMMO00] S. Guha, N. Mishra, R. Motwani, and L. O'callahan. Clustering Data Streams. *Foundations of Computer Science (FOCS)*, September 2000.

[JW93] Gabriel Jakobson and Mark D. Weissman. Alarm Correlation. *IEEE Network Magazine*, Nov 1993.

[YKM+96] Shaula A. Yemini, Shmuel Kliger, Eyal Mozes, Yechiam Yemini, and David Ohsie. High Speed and Robust Event Correlation. *IEEE Communications Magazine*, May 1996.