# Fault localization in Communication Networks

Vijay Akkineni
Department of Computer Science and Engineering
Georgia State University
vakkineni1@student.gsu.edu

## Abstract

This paper is about fault localization techniques in communication networks worked for Phd qualifiers examination. Fault Localization is an important aspect of network fault management and is a process of deducing the source of a failure from the set of observed indicatations. It has been an important research area in the field of networking both communication networks and wireless sensor networks. As commuication networks grow in size and complexity it has been imposing new set of requirements on fault localization. Despite the amount of research done in this field we can still argue that it is an area of open for research. The paper essentially disucsses the work that has been done in this field in the pas few years with emphasis on the both the papers given for the examination.

## Categories and Subject Descriptors

H.4 [**Communication Networks, Sensor Network Applications**]: Miscellaneous

## Keywords

Fault Localization, Communication Networks, Root Cause Analysis, Causal Inference

## 1 Introduction

Fault diagnosis is an important part of networking. Faults are unavoidable in communication systems but their quick detection and isolation and repair is critical for the robustness, reliability and health of the system. When the networks get large and cumbersome automatic fault diagnosis and fault management is a crucial aspect.

A basic taxonomy in this field is mentioned below.

Event, defined as an exceptional condition occurring in the operation of hardware or software of a managed network, is a central concept pertaining to fault diagnosis.

Faults (also referred to as problems or root causes) constitute a class of network events that can cause other events but are not themselves caused by other events. Faults may be classified as: (1) permanent, (2) intermittent, and (3) transient. A permanent fault exists in a network until a repair action is taken. Intermittent faults occur on a discontinuous and periodic basis causing a degradation of service for short periods of time. However, frequently re-occurring intermittent faults significantly jeopardize service performance. Transient faults cause a temporary and minor degradation of service.

Error(Failure) is defined as a discrepancy between a computed, observed, or measured value or condition and a true, specified, or theoretically correct value or condition. Error is a consequence of a fault. Faults may or may not cause one or more errors. Errors may cause deviation of a delivered service from the specified service, which is visible to the outside world. Errors do not need to be directly corrected, and in many cases they are not visible externally. However, an error in a network device or software may cause a malfunctioning of dependent network devices or software. Thus, errors may propagate within the network causing failures of faultless hardware or software.

Symptoms are external manifestations of failures. They are observed as alarmsnotifications of a potential failure. These notifications may originate from management agents via management protocol messages (SNMP trap and CMIP EVENT-REPORT), from management systems that monitor the network status, e.g., using command ping, system logfiles or character streams sent by external equipments.

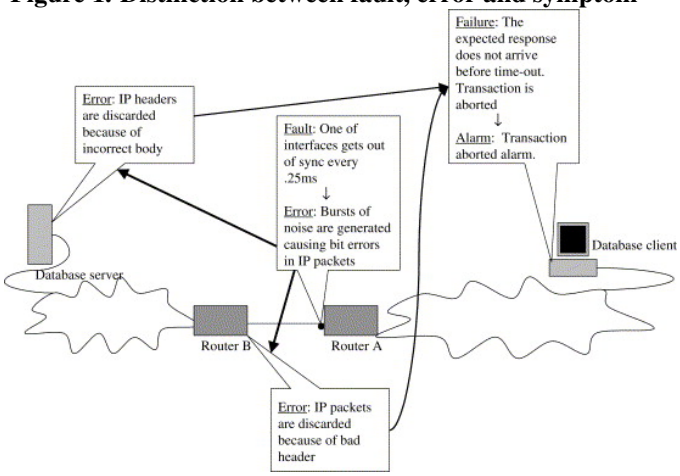
Figure 1 shows the concepts described above in an end to end perspective.

The process of fault diagnosis usually involves three steps as cited in [1]:

- Fault detection, a process of capturing on-line indications of network disorder provided by malfunctioning devices or fault detection agents in the form of alarms.
- Fault localization (also referred to as fault isolation, alarm/event correlation, and root cause analysis) is a set of observed fault indications is analyzed to find an explanation of the alarms.
- Testing is a process that, given a number of possible explanations, determines the actual faults.
- Fault Correction, by which we mean not only to diagnose, but also to repair all faulty components within a

network (This includes the testing component).

**Figure 1. Distinction between fault, error and symptom**




The difficulty in the fault localization process arise from the ambiguity of the observed set of alarms as same alarm can be generated from multiple different faults and multiple alarms can correlate back to the same fault. The incompleteness of the alarm stems from the fact that the alarm doesn't have all the information or there is a loss of alarm. Inconsistency among the observed alarms results from device perception of the fault is different form device to device.

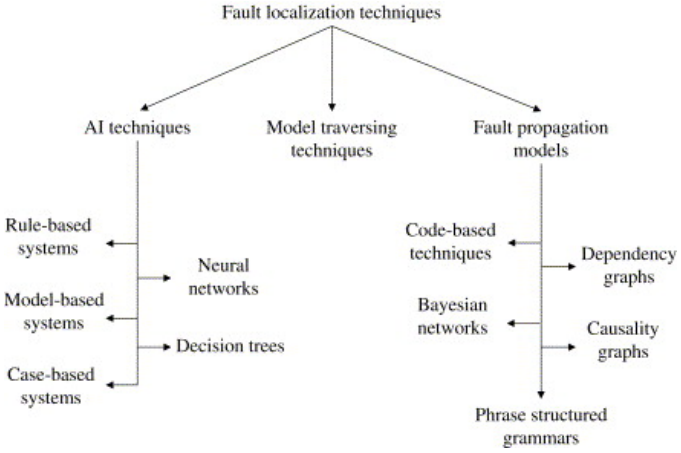
A set of alarms generated by a fault may depend on many factors such as dependencies among network devices, current configurations, services in use since fault occurrence, presence of other faults, values of other network parameters, etc. Due to this non-determinism the system knowledge may be subject to inaccuracy and inconsistency. Fault evidence may also be inaccurate because of spurious alarms, which are generated by transient problems or as a result of overly sensitive fault detection mechanisms. When spurious symptoms may occur, the management system may not be sure which observed alarms should be taken into account in the fault localization process. Event management systems should identify and eliminate multiple simultaneous related or unrelated root causes.

In large networks distributed fault localization process should be performed in distributed fashion. Many researches [2] and [3] have concluded that distributed fault localization using a set of event management nodes is a better approach than centralized process. The complexity arised from the nature of error propogation, they propogate either horizontally to the peers or vertically from bottom layer to upper layers or afecting higher level services. The errors also propogate from one management domain to a different management domain making the process even difficult. Therefore the distribution of complexity to distributed fault localization processes and inferring form the collective process makes the process computationally feasible and efficient.

An alarm can insinuate different type of faults that occurred in different communication devices where in fault localization process may not comeup with a definitive answer. Few approaches that will be discussed in this paper combine fault localization with testing and fault ocrrection process to validate the hypothesis. Therefore there should be some kind of optimality measure or confidence measure that should be employed in measuring the hypothesis that the localization process came up with and it could include the lowest cost or min failure probability or some heuristic function which optimizes the process of validating the hypothesis.

Numerous works have been proposed on fault localization process. The techniques are dervice from different areas of artificial intelligence, graph theory, neural networks, automata theory and many other approaches. Figure 2 broadly classifies the exisiting solutions. The two papers provided for the exam falls into the category of end to end testing scenarios and proababilistic reasoning and derive their roots from bayesian network analysis.

**Figure 2. Classification of fault localization techniques**




## 2 Expert Systems techniques

Most widely used technique in the field of fault localization and diagnosis are expert systems as they try to mimic the actions of human expert. Most expert systems use rule based system as their inference engine. [4],[5] and [6] are examples of expert systems.

The expert systems developed differ in the knowledge they use. Rule based fault localization solely depend on the structure of the knowledge base as rule definition language. In [7] the knowledge base is divided to reusable knowledge modeled as core knowledge and customized knowledge. in [5] the rules are organized as composite events and an intervention of human expert is required to update the event base. The rule based systems can act as a powerful tool to eliminate least likely hypothesis. Although the RBR paradigm is appropriate for problem-solving tasks that are confined and well-understood its limitations are.

- an in-ability to learn from experience.
- Fan inability to deal with novel problems.
- the difxulty of updating the systems to keep up with rapidiy changing domains such as expanding heterogeneous network.

in Model bases expert systems like [6] conditions are ususally accosiated with rules which includes predicates re-

ferring to system model. These predicates test the syetem for existence of relationship among system component. [6] uses correlation tree skeletons describing cause and effect relation ships between event. regardless of what expert system is being used localization process is always driven by inference engine and correaltion rules between events.

[8] and [9] are examples of Case-based reasoning systems. The goals of CBR systems are (i) to learn from experience, (ii) to offer solutions to novel problems based on past experience, and (iii) to avoid expensive maintenance. The basic idea of CBR is to recall, adapt, and execute episodes of former problem-solving in an attempt to deal with a current problem. Former episodes of problem-solving are represented as cases in a case library. When confronted with a new problem, a CBR system retrieves a similar case and tries to adapt the case in an attempt to solve the outstanding problem. The disadvantages of such a system is the time complexity in retrieving a case that is the closest match to the event and the close tailoring of the application to the domain.

In addition to the above mentioned techniques there are other notable techniques are neural networks based approaches [10] [11] and decision tree based approaches [12]. Neural networks have parallel computing architecture and are very fast avoiding bottlenecks which commonly arises from serial processing. But the main disadvantage is that the learning process requires intesive training and in communication networks where all the alarms signatures are not readily available for pattern recognition.

Decision tree aproaches are simple and allows expressive representation of expert knowledge howeever they are limited by the dependencies sepcific applications nad degraded accuracy in noisy scenarios [13] [14].

## 3 Model traversing techniques

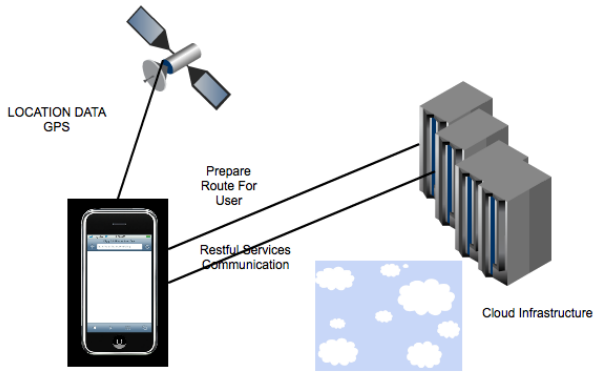
[16], [17], [18] and [19] are all Model Traversing that use formal representation of communication system with clearly marked relationships across network entities.

The design goals for the system are mainly scalability, energy savings, sensor failover and operating system independence. To achieve operating system independence, the main aim was to write code once and deploy it to various mobile operating systems. Open source Titanium API is a platform for developing mobile solutions using web technologies. Titanium allows developers to write application code in Javascript language and API would invoke native API provided by the operating system to access functioanlity realted to geoloation, accelerometer and maps. Project was started off using Titanium API and later realized that the Maps API provided by the titanium doesnt meet the requirements of the project. As custom overlays on the Maps are not provided by titanium api on IOS. Currently Titanium is used only for android based devices and IOS application is developed in native language of Objective-C. The difficulty in stargetting multiple mobile operating systems is in the diference between the API's provided by different sensors and the differences in the mapping solutions used by Android and IOS.

The scalability design goal is acheived by a cloud backing to the mobile solution. The cloud infrastructure chosen is Amazon EC2 and Heroku. Amazon EC2 is the elastic compute cloud provided by Amazon which allows users to rent virtual computers on which they can run their applications. EC2 allows a scalable deployment of an application by providing user the capability to create a machine instance of choice and be able to start, stop or terminate the instance and also gives the capability to scale up the number of instances as needed and charges on per hour basis. Since we can scale the number of nodes hence the term elastic. Heroku is another cloud provider on which much of the development work was done and the reason was for the cost of cloud provider. Heroku provided 750 hours of free development time on their cloud to new users.

The energy savings mechanism was included in polling the sensors for location information. Energy consumption significantly varies based on the sensor that device is using to locate the device. Sensor failover is handled primarly by the host of sensors like AGPS (Assisted Geographical Positioning System), Wi-Fi, Cellular Location and Digital Compass. Much of the smartphones in the market are provided by these basic sensors to assist in geolocation.

**Figure 3. System Architecture**




### 3.1 Server Architecture

The interaction beetween the mobile client and server is carried over the webservices which are designed in a restful manner. The restful services are deployed on a NodeJs platform that is built on Chrome's Javascript Runtime for easily building fast scalable network applications. NodeJs uses an event-driven, non-blocking I/O model that makes it lightweight, effcient and perfect for data-intensive real time applications that run accross distributed devices. Several protocols as HTTP and TCP/IP are built into the node platform. The REST Webservice provides a base url to invoke for a resource (eg: http://trafficanalyzer.gsu.edu/user/123456/updatelocation). When the rest webservice is invoked query data is passed as HTTP body content and is used by the webservice proxy. HTTP post, get, put, delete methods are analogous to create, read, update and delete of a resource. The data passed from the mobile client to the server would be of JSON format example(”facebookid”:1432214523423,”firstname”:”vijay”,

”lastname”:”akkineni”,”latitiude”: 34.304567,”longitude”:-84.056784,”routeinfo”:[binarydata]).

The data passed in via restful urls are parsed and saved in the NoSql Database HBase. HBase is an open source non relational, distributed database. HBase is not a replacement for the traditional database but it is a column oriented database management system that runs on top of Hadoop file system. HBase applications are written in mapReduce style. Hbase system has a set of tables and tables must have aprimary key which will be used to access the data and supports querying only on the primary key. Traffic Analyzer app uses the Facebook social login identity as the unique primary key for every user and also stores the route information in the column families. Installing Hbase and running it on cloud environment has been a challenge to me. To achieve this an Amazom machine instance was created using Hbase installed on the image and the image is provisioned on two nodes. One of the nodes acts as primary node and the other secondary node as HBase needs a master node for house keeping. Since EC2 instances are paid services, as soon as we stop and start the server IP address of the server changes. This issue was resolved by running a DNS server to provide a name service to the instances so that we can use names to resolve rather than IP address.

## 3.2 Client Architecture

The critical component of *Traffic Analyzer* is the core location component to identify the users geo location. Apple IOS provides CoreLocation API for communicating with device hardware to get the users location. Core location supports magnetometer, direction in which the device is pointing and also whether the device is moving also known as its course. The API has several classes to handle the heading and course information in its direction-related API calls.

The key aspect of Traffic Analyzer is to detect the hardware support and act accordingly on applications behaviour. However, location framework does not supports at the API level the availability of hardware, instead the hardware requirements must be specified at the application level before deploying onto the device. The hardware of interest for the requirements are AGPS, Compass, WiFi, Cellular Reception.

The Mapping needs of the application are met by the Mapkit and Google maps frameworks of IOS and Android respectively. These frameworks allows us to embed maps on the application, provide support to add annotations and overlays on top of it, show users location on the map and tracking modes.

## 3.3 Application Functionality

The location framework is an abstraction on top of three main methods of geolocation. The least accuracy level is obtained from the cellular network triangulation method to locate the user and provides a position upto 12km and error can be reduced to 2-3 km based on the closest tower. The next level is provided by the Wifi. This is very precise for about 100m, however the user has to be connected to a wireless hotspot. Finally the highest level of accuracy is provided by GPS and has the error less that 40m. The cost and energy consumption also varies linearly with the decrease of error component in the above hardware. The two parameters that are used int he application are *distanceFilter* and *desiredAccuracy*. These parameters lets the application know on how frequently it will recieve location updates and how accurate the updated readings are.

**Table 1. Desired Accuracy Values for Traffic Analyzer**

| Constant Value | Definition |
|---|---|
| kCLLocationAccuracyBestForNavigation | Standard accuracy intended for navigational apps. |
| kCLLocationAccuracyBest | Use highest accuracy available. |
| kCLLocationAccuracyNearestTenMeters | 10 meters accuracy. |
| kCLLocationAccuracyHundredMeters | 100 meters accuracy. |
| kCLLocationAccuracyKilometer | Accuracy upto 1km. |
| kCLLocationAccuracyThreeKilometers | Accuracy upto 3km. |

In Traffic Analyzer saving energy is a primary deisgn goal and consumption is reduced based on the hardware device we choose for getting the location info. Our application checks the battery level and when the energy levels of the battery depletes to 30% of the total energy it fallsback to the least accurate mode of wifi or cell provider.

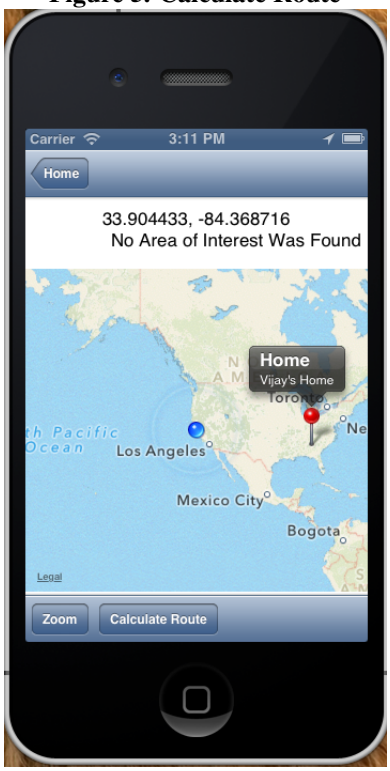
The application starts by user entering the target address where he is headed and a route planning is performed. Once the input address is forward geocoded so that address is translated to latitude and longitude. Once the driving directions and user route plan is finalized by selecting one of the routes from the driving directions. The figure 2,3 and 4 describes this behaviour. Planning will now be performed to decide when to perform polling and where. Polling here refers to the querying of the device GPS to fetch the location information. Once querying the device hardware for the location info, this information is now transferred to the server using the restful services that are exposed.

The polling logic currently is very basic and would need further refinement based on statistics. Polling is achieved by using dispatch async that would lets us call a thread and dispatch after can be used in conjuction to achieve this functionality. The current logic is based upon speed and distance filter. NSTimer is an useful function provided by IOS which enables us to configure timer based on an interval. This timer is used to trigger the dispatch queue threads that perform atomic operations of updating the location information in the server.

- If the destination is above 5 miles and the current speed of the vehicle is above 50 MPH polling is performed only every 15 secs and the distance filter is increased to 500m such that we do not have to perform querying often.
- If the user is travellign at speeds less than 10MPH

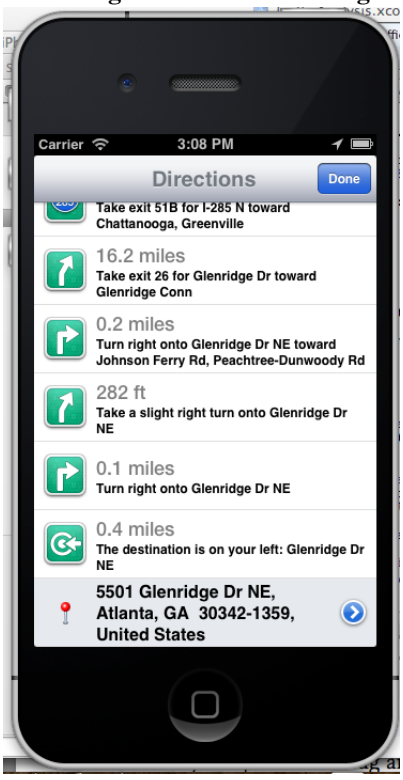
**Figure 4. Address Input**


**Figure 5. Calculate Route**




polling interval is set to 15secs and distance filter is set to 20m.

**Figure 6. Driving Directions for Polling Planning**




- If the user speed is in between 15-50, then polling interval is set to 5, 8 and 12 secs respectively for 20mph,30mph,40mph.

The coverage area of the mobile is limitied so in this scenario the GPS would be able to read the position but with the lack of coverage data provider the server calls would fail. To overcome this kind of failure a SQL Lite database on the mobile is used to store temporary information that needs to be passed to the server. The structure of the table in SQL Lite database is very similar to the information that is passed in to the restful services. The table is called TEMP_USER_LOC_INFO which has the attribtues of latitude, longitude and other route related info of speed, direction and etc.

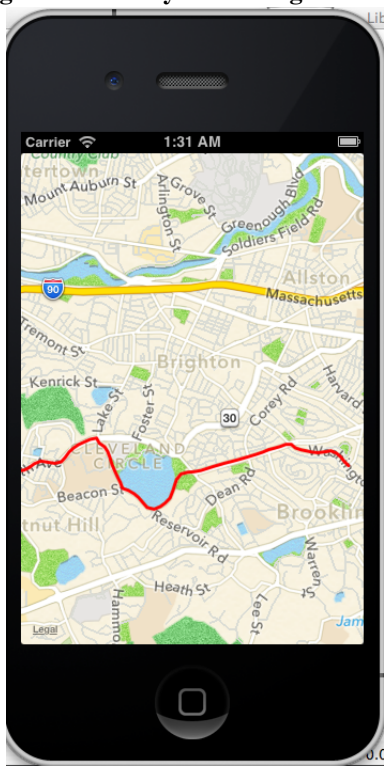
The traffic rendering part on the maps is achieved via overlay's on the map. On the load of the screen to view traffic a webservice pulls all the coordinates on the route intersection and draws overlay on top of the view. The traffic is rendered as red, yellow and green paths along the coordinates.

## 3.4 Kalman Filtering

Kalman Filter is also known as linear quadratic estimation is an algorithm used to measure a series of measurements observed over the time and containing noise (random variations) and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone.

The state transition equation for velocity would be $V_n = V_{n-1} + w_n$. W represents the noise in the system. Next we prepare kalman filter inputs and constants and the the algo-

**Figure 7. Overlay Rendering of Traffic**




rithm proceeds in the two steps in the first step it predicts the next data for velocity and in the second step thee estimates are updated with the weighted average of the actual measurement involved. Kalman filter is implemented on the server side however there are still issues with implementation of kalman filter on how to run it on demand and update the data variables.

## 4 Timeline of Project Proposal

Timeline:

- Research on infrastructure to generate application on multiple smart phones. ( 2 weeks)
- Enable Facebook Social Network based login for identity purpose. (1 week)
- Setting up cloud infrastructure and using a Map Reduced based Hbase NoSQL to store User data. (2 weeks)
- Integrating communication from the Application to the Cloud. (1 week)
- Rendering overlay of vehicles and traffic on the Map. ( 2 weeks)
- Experimenting with Kalman Filtering to predict state. (1 week)

## 5 Conclusions and Future Work

The Traffic Analyzer project was able to successfully track user movement and store them in a databse and render the traffic pattern on the maps. The project was able to meet the proposed requirements. There are still a lot of areas to improve in the project. The current polling based mechanism is very primitive and polling can still be improved with optimization methodologies and statistics driven approaches and the kalman filtering is currently is not producing accurate results. Kalman filtering can smooth data in the abscence of data. Further evaluation is needed in mapping solutions whcih provide ease of drwaing overlays and integration with various other data sources apart from the sensors should be researched. Project repo can be found at https://github.com/akkinenivijay/TrafficAnalyzer

## 6 Evaluation

My evaluation criterion for the project depends on the goals I desired to achieve. The goal for me is to achieve A in this course, upon the delivery of the Mobile App communicating with cloud infrastructure and rendering of the data on Google Maps. This is the brunt of the project and requires significant effort. Enabling social integration and kalman filtering are additional characteristics. Overall I am happy with the learning outcome from this project as I learned both Android, IOS Systems and Map API's.

## A References

[1] I. Katzela Fault diagnosis in telecommunications networks, Ph.D. Thesis. School of Arts and Sciences, Columbia University, New York, 1996.

[2] I. Katzela, A.T. Bouloutas, S.B. Calo Centralized vs distributed fault localization. A.S. Sethi, F. Faure-Vincent, Y. Raynaud (Eds.), Integrated Network Management IV, Chapman and Hall, London (1995), pp. 250263.

[3] S.A. Yemini, S. Kliger, E. Mozes, Y. Yemini, D. Ohsie. High speed and robust event correlation. IEEE Communications Magazine, 34 (5) (1996), pp. 8290.

[4] Peng Wu, Rajiv Bhatnagar, Lennie Epshtein, Malini Bhandaru, Zhongwen Shi. Alarm Correlation Engine (ACE). GTE Laboratories Incorporated.

[5] G. Liu, A.K. Mok, E.J. Yang. Composite events for network event correlation - JECTOR. Integrated Network Management VI, IEEE (1999), pp. 247260.

[6] Y.A. Nygate. ECXPERT: Event correlation using rule and object based techniques. Integrated Network Management IV, Chapman and Hall, London (1995), pp. 278289.

[7] K.-W.E. Lor. A network diagnostic expert system for Acculink multiplexers based on a general network diagnostic scheme. Integrated Network Management III, North-Holland, Amsterdam (1993), pp. 659669.

[8] L. Lewis. A case-based reasoning approach to the resolution of faults in communications networks. Integrated Network Management III, North-Holland, Amsterdam (1993).

[9] R.D. Gardner, D.A. Harle. Methods and systems for alarm correlation. Proc. of GLOBECOM, London, UK, November (1996), pp. 136140.

[10] R.D. Gardner, D.A. Harle. Alarm correlation and network fault resolution using the Kohonen self-organizing

map. Proc. of IEEE GLOBECOM, Toronto, Canada, September (1997).

[11] R.D. Gardner, D.A. Harle. Pattern discovery and specification techniques for alarm correlation. NOMS98, Proc. Network Operation and Management Symposium, New Orleans, LA (1998), pp. 713722.

[12] G.D. Rodosek, T. Kaiser. Intelligent assistant: User-guided fault localization. Ninth Intl Workshop on Distributed Systems: Operations and Management, University of Delaware, Newark, DE, October (1998), pp. 119129.

[13] S. Russell. Machine learning. Artificial Intelligence, Handbook of Perception and Cognition (second ed.), Academic Press, New York (1996), pp. 89133 (Chapter 4).

[14] Daphne Koller, Nir Friedman Probabilistic Graphical Models. MIT Press.

[15] M. Steinder and A.S Sethi. The present and future of event correlation: A need for end to end service fault localization. In N. Callaos et al. World Multi-Conf. Systemics, Cybernetics and Informatics, Vol XII, Orlando, FL, 2001. pp. 124-129.

[16] S. Ktker. A modeling framework for integrated distributed systems fault management. In C. Popien (Ed.), Proc. IFIP/IEEE Internat. Conference on Distributed Platforms, pp. 187198, Dresden, Germany, 1996.

[17] S. Ktker, K. Geihs. A generic model for fault isolation in integrated management systems. Journal of Network and Systems Management, 5 (2) (1997), pp. 109130.

[18] S. Ktker, M. Paterok. Fault isolation and event correlation for integrated fault management. Integrated Network Management V, Chapman and Hall, London (1997), pp. 583596.

[19] B. Gruschke. Integrated event management: Event correlation using dependency graphs. A.S. Sethi (Ed.), Ninth Internat. Workshop on Distributed Systems: Operations and Management, University of Delaware, Newark, DE, October (1998), pp. 130141.