23rd International Symposium on Fault-Tolerant Computing: Toulouse, France. 22-24 June 1993.

Detection and Discrimination of Injected Network Faults

Roy A. Maxion and Robert T. Olszewski

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
InterNet: Maxion@CS.CMU.EDU

Abstract

Six hundred faults were induced by injection into five live campus networks at Carnegie Mellon University in order to determine whether or not particular network faults have unique signatures as determined by out-of-band monitoring instrumentation. If unique signatures span networks, then the monitoring instrumentation can be used to diagnose network faults, or distinguish among fault classes, without human intervention, using machine-generated diagnostic decision rules. This would be especially useful in large, unmanned systems in which the occurrence of novel or unanticipated faults could be catastrophic. Results indicate that significant accuracy in automated detection and discrimination among fault types can be obtained using anomaly signatures as described here.

1. Objective

Fault detection and diagnosis in noisy, nonlinear, nonstationary domains is intrinsically hard. This is complicated further when one's understanding of the particular domain is incomplete, thus making modeling difficult. Examples of such domains, whose dynamic characteristics change over time, are chemical plants (catalyst activity decays with use), semiconductor fabrication processes (plasma etch characteristics change due to erosion and accumulated contamination) and network communications (traffic and configuration conditions drift with time). Perhaps because such domains are so difficult to track and understand, reliable fault detection and diagnosis techniques are badly needed.

It is common that in the environments described, no training examples or training sets are available, particularly for unanticipated faults. This is largely because these environments are incompletely understood, due to their complexity and dynamic character. Moreover, their missions tend to change frequently, as when new chemicals or semiconductors are brought to manufacture, or when new applications, software or machinery are placed in service on a network. The present work is directed at understanding how to acquire enough information from the environment itself to be able to formulate a useful, adaptable fault detection and diagnosis system.

Although the present work does in fact employ training data, it does so in the interest of calibrating the results obtained from an experimental detection and diagnostic system designed specifically to accommodate noisy, nonstationary, nonspecific domains. The system generalizes by virtue of its log analysis capabilities; all monitored data and events are recorded in log files. These files are processed by the system, resulting in testable and reproducible detections and diagnoses of anomalous conditions. Any monitored process or device can be used to populate the logs with data.

The specific objective of the present work is to conduct a designed experiment to test the detection and diagnosis capabilities of a system for handling faults in local area networks. Networks were selected as a test domain because their operating characteristics include nonlinear, nonstationary dynamic behavior. The experiment uses automated injection techniques to induce carefully calibrated fault conditions into multiple, live networks. These faults may manifest differently on different networks. The experiment aims to use monitored results from the injections in determining whether or not it is possible to construct reliable diagnostic decision rules, spanning all injected networks, that will discriminate successfully among various fault conditions.

If the experiment is successful, suggesting that fault-specific signatures are consistent across network domains, then out-of-band monitoring instruments can be used to identify network faults or fault classes, without human intervention, using machine-generated diagnostic decision rules. This would be especially useful in large, unmanned systems in which the occurrence of unanticipated faults could be catastrophic.

2. Background and related work

A long-term goal is to develop diagnostic systems that, like people, can ascertain from their environments the necessary relationships among environmental stimuli for ensuring success in their missions without external aid. The word "mission" can be interpreted in several ways: maintain equilibrium or homeostasis; compensate for the effects of external disturbances; keep a system operating under unanticipated conditions that may affect

Roy A. Maxion and Robert T. Olszewski. Detection and Discrimination of Injected Network Faults. In *The Twenty-Third International Symposium on Fault-Tolerant Computing,* pages 198-207. IEEE Computer Society Press, 1993.

its ability to deliver reliably dependable service. Implied in all these is the idea of adaptation to changes in operational environments. One problem encountered in evolving environments is that the definition of a fault may change, depending on local circumstances. Based on static measures, what constitutes a fault under some conditions may not constitute a fault under other (often unpredictable) conditions in a different time or place. This suggests that a diagnostic system must find a way to accommodate to these changing situations.

The present work builds on foundations established earlier by Maxion and Feather. Maxion [11] established the learning mechanism by which adaptive stability could be achieved in environments whose operating characteristics drift over time. Maxion [12] also showed that process anomalies could be detected reliably in noisy communication environments like Ethernet networks. Maxion and Feather [13] showed that anomalous network events such as broadcasts and jabbers could be detected using these anomaly detectors, but this was done by empirical observation, rather than by designed study. Feather [5] replicated this work, and showed that vectors of measured features could be indicative of network faults, although again no designed experiment was done. The current work extends earlier results by performing a designed experiment to determine whether or not faults can be represented reliably by vectors of measured parameters undergoing continuous, environmentally-driven adaptation, and whether or not such vectors are sufficiently consistent under different environmental conditions to be classified reliably by a decision rule applied to the vector features. One advantage of this technique is that adaptation to new or changing environments is automatic.

Other work in diagnosis addresses related, but not identical, issues. In contrast to the present technique, many approaches have relied largely on preconceived, static fault definitions in predominantly stable environments, dependent to varying degrees on characteristics of particular systems. One example is that of Lin and Siewiorek [9] who used system-dependent trend analysis rules in measuring system behaviors to suggest that increasing rates of intermittent faultiness are predictive of hard failure. In a similar approach to system-level diagnosis, Tsao [16] used heuristics that relied on the relative timing of fault detections at different points in a system's architectural hierarchy. Signature classification was not a part of these approaches, and the particular heuristics used may have been valid only for the architectures being used, needing to be invented anew for alternative architectures. The adaptive distributed system-level diagnosis (DSD) work of Bianchini et al. [3] [2] has addressed network diagnosis by detecting faulty nodes in a network. The method relies on signals transmitted among the nodes to determine the up/down operational status of a given node. The node itself could be operational, but still faulty in terms of transmitting destructive network traffic, which the DSD algorithm would never see. It is exactly this kind of situation that the present work avoids, by detecting anomalous conditions in the network traffic environment itself, not the originating nodes. Iyer et al. [6] have devised an approach that takes raw error logs containing a single entry for each error, and produces a list of symptoms that characterize persistent errors. Lee et al. [8], in an analysis of event logs, identified errors on the basis of knowledge of the architectural and operational characteristics of systems measured. Similarly, Thearling and Iyer [15] developed a diagnostic reasoning technique which operates on the basis of observed erroneous behavior and system structure.

3. Faults, anomalies and signatures

Historically, "fault" and "failure" have been defined in several ways. Maxion and Feather [13] noted the evolution of the definition of "fault" from Avizienis [1] (the deviation of one or more logic values ... from their design-specified values) through Siewiorek and Swarz [14] (a physical change in hardware) to Laprie's [7] "the cause of an error". Perhaps the least context-sensitive definition comes from Webster's 3rd International Unabridged Dictionary: a fault is "a defect in quality" and a failure is a "lack of satisfactory performance or effect; to fail is *to disappoint the expectations of*" (emphasis added).

The clear sense one gains here is that a failure is a departure from expected conditions. Expectations can be set either by specification or by experience, neither of which is guaranteed to be complete. In real-world environments, expected conditions are normal conditions, and hence there is some motivation for casting deviations from normalcy as failures.

In the present work a fault is regarded as a departure from expected operating conditions -- it produces an anomaly in the observed operating environment. Many faults have characteristic suites of symptoms, sometimes called syndromes or signatures. We want to see if network faults (1) can be detected reliably by noting performance anomalies; and (2) can be discriminated by virtue of their signatures. If signature discrimination is possible, then simple, machine-generated pattern-recognition filters can be used to detect and diagnose network faults in real time. Similar techniques can be applied in many other domains.

Signatures are often thought of in terms of absolute measures on various parameters of a stimulus signal. For network traffic phenomena this would mean absolute measures of raw packet traffic or raw collision counts, for example. Because different networks operate normally at different mean levels, an absolute collision count of 200 per minute on one network might be acceptably normal, while on another network, perhaps less utilized, the same count of 200 would be unacceptably high. Hence, it is difficult to compare measurements between different network environments. Maxion and Feather [13] have discussed the implications of normalcy and faulty behavior previously.

Roy A. Maxion and Robert T. Olszewski. Detection and Discrimination of Injected Network Faults. In *The Twenty-Third International Symposium on Fault-Tolerant Computing,* pages 198-207. IEEE Computer Society Press, 1993.

To facilitate a standard of comparison among different network environments, the measure used was not raw counts, but rather levels of anomalous conditions. Therefore, a count of 200 collisions per minute on one network might be considered normal, while the same count on another network would be consider out of normal range, or anomalous. The comparison, then, is a measure of the extent to which a network operates within its normal range on any of a number of dimensions. On any dimension a network may measure normal or anomalous. The signature uses these indications of normal or abnormal, instead of raw measures. While absolute raw measures may change, anomalies do not. Maxion [12] has discussed how a monitoring system can adapt its concepts of anomalous behavior, even when the observed environment is changing over time.

4. Experimental environment

Given that the objective is to test the adequacy of anomaly signatures under various operating conditions, there are many experimental environments that would be suitable. Since much of our previous work has been conducted in the network environment, and networks exhibit the characteristics necessary for testing the idea of anomaly signatures (noise, trend, dynamic behavior, disparate environments, etc.), the CMU campus network was selected as a real-world environment for experimentation.

Monitoring and diagnosis of networks is tremendously challenging. Special equipment is needed, continuous, error-free monitoring is necessary, and the operating conditions of networks themselves practically defy scrutiny. It is perhaps less the goal to show that the proposed technique works for networks than it is to show that it works at all. If it works in networks, it should also work in less challenging circumstances.

4.1. CMU network configuration

The CMU Computer Science network is configured with a Cisco AGS+ intelligent bridge/router acting exclusively in bridging mode as a common backbone for 18 Ethernet networks, 13 of which are in use. Five of these 13 networks were employed in the present injection and monitoring study. The machine population of the network is roughly 1300 stations of various kinds, including servers. The population must be estimated at any given time because it changes rapidly, often from one day to the next.

4.2. Performance of networked computers

The computers on each monitored network were classified into two groups: low and high performance. Two groups were used, instead of three, to avoid splitting hairs in attempting to classify something as medium performance, rather than high or low, in a three-way classification scheme. The classification was based primarily on a machine's MIPS rating as an indication of its ability to saturate the network. It should be noted that high-end machines are often used as servers, which are typically responsible for more network bandwidth than clients. Symmetrically, the lower-end machines are most often used as client machines. High-performance machines employed as clients only infrequently monopolize network bandwidth in large data transfers. There were 419 client machines and 20 server machines included in the study, as shown in Table 4-1.

Machines typical of each type are:

- Low -- Sun-3, IBM RT, NeXT, Encore Multimax, microVAX/VAX, Motorola MC68xxx, Sequent Balance, Gateway, Toshiba 5200, IBM L40SX, Intel i386, Omron, Ariel, Macintosh, Olivetti M380, Kinetics, Sony NWS-1250, TI Explorer.
- High -- DEC 2100-5000, Sun-4, i486/66, HP 720/730, IBM RS6000, Alpha, Iris.

Network	System Performance Level	Clients	Servers	Total
Network 1/5FL	low	57	0	57
	high	57	1	58
				115
Network 2/3PU	low	76	2	78
	high	71	1	72
				150
Network 3/3PR	low	11	6	17
	high	14	9	23
				40
Network 4/7FL	low	18	0	18
	high	34	0	34
				52
Network 5/DOH	low	45	0	45
	high	36	1	37
				82

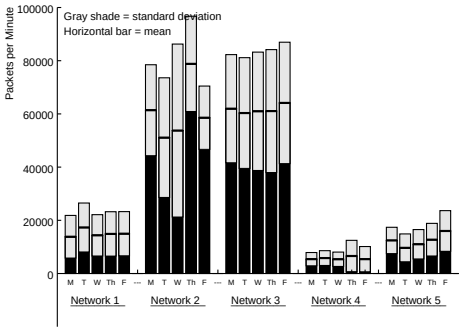
Table 4-1: Breakdown of machine performance classes, high and low, on each monitored network. 5FL, 3PU, etc. are codes for network names. Total population is 439 systems.

4.3. Traffic characteristics

Different networks exhibit different behaviors. Non-uniform network behavior is one of the conditions under scrutiny in this work, since there is an expectation that individual network characteristics will influence fault detection and/or diagnosis. Five networks were selected for the study on the basis of their availability and their operating characteristics. One of the networks had a low-performance client population and no servers (network 4). One of the networks had relatively few clients, but many servers (network 3). Network 2 had a large client population, including many high-performance clients, and a moderate server population. Network 1 has over a hundred clients, evenly split between low- and high-performance machines, and only one server. Network 5 has a slightly lower population,

Roy A. Maxion and Robert T. Olszewski. Detection and Discrimination of Injected Network Faults. In *The Twenty-Third International Symposium on Fault-Tolerant Computing,* pages 198-207. IEEE Computer Society Press, 1993.

but its clients are engaged in graphics and virtual reality research, making them heavier network users than might otherwise be expected. Figure 4-1 shows the mean and standard deviation performance in packet traffic per minute for each of the five networks over the same five-day period. Activities that are typical of network usage at CMU are: electronic mail, AFS/NFS/RFS, X, telnet, and backup (which has the highest average network usage of all these activities). Most traffic is tcp/ip (udp).

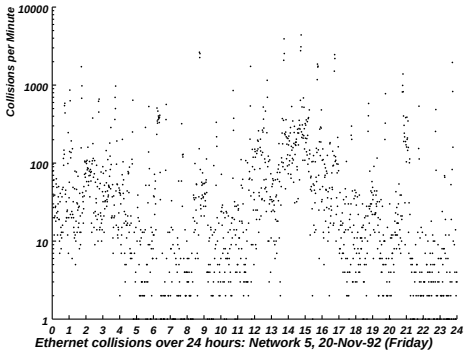


Mean daily packet traffic, November 9-13, 1992.

Figure 4-1: Mean and standard deviation of daily packet traffic for five different monitored networks. There are significant differences in the behaviors of the several networks, that could be expected to influence the detection of faults and the ability to discriminate among them.

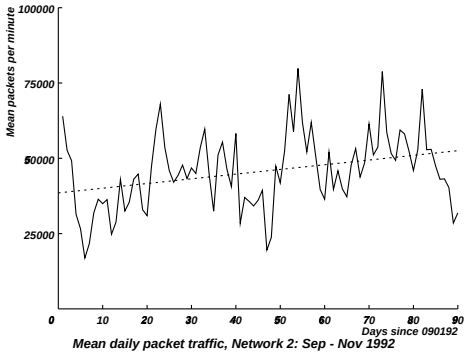
To provide some idea of the kind of data being investigated, and also an indication of the difficulty of doing so, Figure 4-2 shows an example of a typical network day on one of the quieter networks, network 5. It is interesting to note that all of the injected faults for this network were correctly detected for the day shown.

One characteristic required for the study was nonstationarity, or trend. If the characteristics of the stochastic process that generated the time-series network behavior are not invariant with respect to time (i.e., the process is nonstationary), it will often be difficult to represent the time series over past and future time intervals by a simple model. Trend, loosely defined as a long-term change in the mean, is typically measured in terms of the slope of the regression line plotted over the designated time period. A regression analysis of the mean daily packet traffic for network 2 showed significant positive trend at the .0001 level. The regression equation is y = 38516.5 + 155.7x. The data are graphed in Figure 4-3 with the regression trend line superimposed. Other networks analyzed showed similar positive trend at similar significance levels.



Ethernet collisions over 24 hours: Network 5, 20-Nov-92 (Friday)

Figure 4-2: Scattergram showing raw collision data containing 24 fault injections, all detected correctly.



Mean daily packet traffic, Network 2: Sep - Nov 1992

Figure 4-3: Ninety-day mean daily packet traffic (network 2) showing positive trend (dotted line) significant at the .0001 level.

5. Injected faults

Five faults were selected for injection testing. Each fault is representative of typical network faults that could impair or disable network performance. The following fault conditions were selected for injection, based on their propensity for natural occurrence, and on the abilities of our instruments to induce and monitor them faithfully and repeatably. These fault types accounted for 76% of confirmed faults over an observation period of two years on the Computer Science networks at Carnegie Mellon University.

Pseudo-runt flood -- A runt packet is one which violates the Ethernet standard by being smaller than the

Roy A. Maxion and Robert T. Olszewski. Detection and Discrimination of Injected Network Faults. In *The Twenty-Third International Symposium on Fault-Tolerant Computing,* pages 198-207. IEEE Computer Society Press, 1993.

60-byte specified minimum size. Because our instrumentation lacks the capability to detect true runts, 60-byte packets were employed as pseudo-runts. This did not affect detection and discrimination. High packet rates are characteristic of runt floods, resulting in resource contention problems.

Network paging -- This condition occurs when a diskless workstation runs a process too large for its physical memory. Memory swaps are then done over the network to a file server. When conducted over even moderately extended periods, network paging can cause significant latency due to bandwidth limitations.

Bad frame-check sequence -- This condition occurs when the frame-check sequence on a packet has been corrupted or incorrectly computed by the receiving station; this necessitates a retransmission, which increases traffic moderately on the network.

Jabber -- This condition is caused by excessive transmission of oversized packets, i.e., packets longer than the protocol-specified 1518-byte limit. The condition results in extreme loss of bandwidth.

Broadcast storm -- This condition causes host response time to become slow, and network utilization to be very high. It is often caused by flawed protocols, software errors or flawed configurations that overuse the Ethernet broadcast address. Broadcast packets are sent to every node on the network. In accordance with the Ethernet protocol, every station is required to read broadcast packets. When there are too many broadcast packets, stations become bogged down in attempting to respond to every packet. (Normally, hosts respond only to packets sent to its own destination, ignoring all other packets.) Because excessive broadcasts would be detrimental to network operations, the broadcast fault was simulated using short packets excluding the broadcast address. For this reason, the broadcast fault feature vector may be similar to, but still discriminable from, the pseudo-runt feature vector (the collision feature in a real broadcast storm would have a high value).

6. Measured data

Fourteen parameters of network performance were measured, as described in items 4-17 below, subject to the limitations of available instrumentation. Measures were taken on two general types of information -- primary and secondary. Primary, direct measures were items 12-17, such as percent utilization and collision count. Secondary, or inferential, measures were address items 4-11, which covary in this experiment with fault-specific performance.

7. Instrumentation

Instrumentation consisted of a fault injection system and two out-of-band hardware monitoring systems.

```
1-  FAULT TYPE
      1 = 60-Byte Pseudo-Runt Storm
      2 = Network Paging
      3 = Bad Frame-Check Sequence
      4 = Jabber
      5 = Broadcast
2-  NETWORK TYPE (1, 2, 3, 4, 5)
3-  INJECTION NUMBER (1-24)
4-  DESTINATION ADDRESS, UNUSUAL ACTIVITY
5-  DESTINATION ADDRESS, INCREASED ACTIVITY
6-  DESTINATION ADDRESS, CEASED ACTIVITY
7-  DESTINATION ADDRESS, SUDDEN APPEARANCE
8-  SOURCE ADDRESS, UNUSUAL ACTIVITY
9-  SOURCE ADDRESS, INCREASED ACTIVITY
10- SOURCE ADDRESS, CEASED ACTIVITY
11- SOURCE ADDRESS, SUDDEN APPEARANCE
12- PERCENT_UTILIZATION
13- PACKET_COUNT
14- COLLISION_COUNT
15- PACKET LENGTH < 63
16- PACKET LENGTH 64-127
17- PACKET LENGTH > 1024
```

Table 6-1: Measured parameters (4-17) which describe anomalies on the network. These elements, together with items 1-3, constitute the features in the anomaly feature vectors.

7.1. Fault injection system

The fault injection system operates in one of two modes. One is to accept a schedule of fault types, network segments and times; at the appointed time, the chosen fault is automatically injected into the selected network. The schedule can extend over many weeks, months, or even years. The second mode is more automated in that the system itself selects the fault types and injection times, thereby constituting a "robo-colleague" who conducts blind experiments. In either mode a log is kept of the faults injected and their injection times. This log can be used after the fact to verify what actually happened. In the blind-experimenter mode, diagnostic systems can be used to detect and identify faults without knowing in advance what the faults are, or when they occurred; diagnoses are later verified against the log. The present experiment was not blind.

The software/hardware system for fault injection consists of a MicroVax workstation which drives the fault-injection software, and an HP 4972A LAN analyzer. The MicroVax communicates with the HP, using the DDCMP protocol. When a fault is to be injected, as determined by the injection schedule, the MicroVax synthesizes the injection code which is then downline loaded onto the HP in real time, after which the HP is remotely commanded to execute the code. Execution thereby causes the HP to generate code-directed traffic patterns and faults, injecting the traffic into the active network. During this period the HP is under remote control by the MicroVax.

Roy A. Maxion and Robert T. Olszewski. Detection and Discrimination of Injected Network Faults. In *The Twenty-Third International Symposium on Fault-Tolerant Computing,* pages 198-207. IEEE Computer Society Press, 1993.

7.2. Monitoring system

The out-of-band monitoring instrumentation consists of two custom pieces of hardware, each controlled by software on the MicroVax. Hardware monitors are attached to each monitored network. One instrument gathers statistics for packet traffic, collisions, percent utilization and uptime. The other instrument gathers data about packet types, packet lengths, packet sources and destinations, etc. All instrumentation is time synchronized. Monitors operate at a 200 ms sample rate, and observations are collapsed into one-minute periods. All data are written to on-line logs in real time. Because the monitoring is done out of band (monitoring instruments do not share the same data path, or band, as the measured device, hence do not influence measured devices by their presence), monitoring has no effect on the parameters being measured.

7.3. Analysis system

The analysis system is completely automated. Monitored data, logged by filename according to data type, time and date, are processed without human intervention as they accumulate. Some exceptions were made at the end of the experiment, because the evaluation of the experimental data demanded certain manipulations that are not part of everyday processing, such as gathering all results into a single file for global assessments. Details of relevant analyses are given in Section 8.3.

8. Methodology and experimental design

8.1. Injection

Five different faults were used. Each fault was injected into each network once an hour for twenty-four hours over the course of a week. Each network received injections for only one day each week, so that the network could recover from the effects of the injections before a new fault would be injected. Table 8-1 shows the schedule of faults injected into each different network.

For each of five weeks a new fault was injected each week. Conducting the experiment over the course of five weeks allowed each network to take its evolutionary course, and facilitated validation of analysis algorithms for adapting to trend. Using five different networks over the course of a week permitted observation of behavioral effects in five different network environments. Conducting the experiment over the course of a 24-hour day allowed faults to be injected across a range of conditions for that day's network.

8.2. Data collection

Monitoring resolution is presently set at 200 ms. Data are collapsed into one-minute epochs, thereby potentially producing 1440 feature vectors per day per network. Over the course of the entire experiment, which was monitored for five weeks plus the preceding eight weeks to permit the adaptive algorithms to stabilize, 468,000 primary observations were taken across the five networks (weekend days excluded), consuming 181.7 megabytes of storage. When the data were expanded for analysis, they occupied even more space, and spanned 2925 files. The bookkeeping alone is a considerable chore.

Fault-1 (60-Byte pseudo-runt storm)					
Week-1	**Net-1**	**Net-2**	**Net-3**	**Net-4**	**Net-5**
	Mon	**Tue**	**Wed**	**Thu**	**Fri**
Fault-2 (Network paging)					
Week-2	**Net-1**	**Net-2**	**Net-3**	**Net-4**	**Net-5**
	Mon	**Tue**	**Wed**	**Thu**	**Fri**
Fault-3 (Bad frame-check sequence)					
Week-3	**Net-1**	**Net-2**	**Net-3**	**Net-4**	**Net-5**
	Mon	**Tue**	**Wed**	**Thu**	**Fri**
Fault-4 (Jabber)					
Week-4	**Net-1**	**Net-2**	**Net-3**	**Net-4**	**Net-5**
	Mon	**Tue**	**Wed**	**Thu**	**Fri**
Fault-5 (Broadcast storm)					
Week-5	**Net-1**	**Net-2**	**Net-3**	**Net-4**	**Net-5**
	Mon	**Tue**	**Wed**	**Thu**	**Fri**

Table 8-1: Schedule of injections for five faults.

Data quality was very good. Missing or corrupt data constituted only .1% of the data. Standard data imputation techniques [10] are employed to recover from these lost data.

8.3. Detection

The raw data are broken out into individual files containing lists of anomalies for each different parameter. Determination of anomalous points is achieved by the adaptive method described by Maxion [12]. Anomalies can take values from -2 to +2, including zero. Zero means that no anomaly was detected. Positive values indicate that monitored conditions differed from expected conditions by a positive value; negative values indicate the converse. The value itself indicates the magnitude of the anomaly; a 1 or -1 depicts an event that is merely surprising, as opposed to a 2 or -2, which depict events that go beyond surprise; they are astonishing. Quantitatively, these levels can be thought of as standard deviations, where 1 maps to 3 standard deviations above expected behavior, and 2 maps to 4 or more. "X" signifies "don't care." A "don't-care" value would be appropriate for a parameter that is not affected by a particular fault, nor does its value help discriminate among different faults.

Each type of data is regarded as a feature. Software is used to coalesce the individual features from separate databases into a database of feature vectors for each monitored time frame. The feature vectors contain all 14 features, plus information strictly for experimental purposes, like fault type, network number and injection sequence number. Table 8-2 shows some of the fea-

Roy A. Maxion and Robert T. Olszewski. Detection and Discrimination of Injected Network Faults. In *The Twenty-Third International Symposium on Fault-Tolerant Computing,* pages 198-207. IEEE Computer Society Press, 1993.

ture vectors for fault 3 injected into network 2. The columns follow the order of variables shown in Table 6-1. Except for the first three columns, each number can be regarded as a measure of the extent of an anomaly on its respective parameter at a particular moment in time.

3	2	1	2	2	0	0	2	2	1	0	0	0	0	0	0	0
3	2	2	2	2	0	0	2	2	0	0	0	0	0	0	0	0
3	2	4	2	2	0	0	2	2	0	0	0	0	0	0	0	0
3	2	5	2	2	1	0	2	2	1	1	0	0	0	0	0	0
3	2	6	2	2	0	0	2	2	0	0	0	0	0	0	0	0
3	2	7	2	2	0	0	2	2	0	0	0	0	X	0	0	0
3	2	8	2	2	0	0	2	2	0	0	0	0	0	0	0	0
3	2	9	2	2	0	0	2	2	0	0	0	0	0	0	0	0
3	2	10	2	2	0	0	2	2	0	0	0	0	X	0	0	1
3	2	11	2	2	0	0	2	2	0	0	0	0	0	0	0	0
3	2	12	2	2	0	1	2	2	0	1	0	0	0	0	0	0

Table 8-2: Feature vectors for fault 3, network 2.

8.4. Discrimination

Diagnosis means to identify a condition by its signs, symptoms or distinguishing characteristics; to discriminate among distinct features. Put another way, certain features are diagnostic of particular conditions. The feature vectors obtained in the injection study are already known to be of five groups or classes -- one class for each fault -- but exactly what features may discriminate them from one another is unknown. Each feature vector was assigned a fault number from 1 to 5, according to its associated injected fault.

What is needed is a decision rule, based on the vectors' discriminating characteristics, for assigning feature vectors of unknown origin to correct fault classes. The decision rule can then be used to discriminate one feature vector, or fault, from another on the basis of the features and values contained in the vector. Such a decision rule would permit automatic discrimination among new occurrences of faults. The rule should, for example, be able to discriminate among the three following feature vectors, assigning each to a fault category. (These three vectors were correctly classified by the decision rule automatically generated in the course of this project. Two of them represent the same fault, but on different networks; the other is a different fault on the same network as the first.)

```
1 1 0 1 1 1 0 1 0 1 0 1 0 0
1 1 0 1 2 2 0 1 0 0 0 1 0 0
1 1 0 1 1 1 0 1 1 1 0 0 0 1
```

The technique used here to do this is called recursive partitioning regression [4], because the procedure forms homogeneous groups (of vectors) by recursive partitioning, or splitting, of the dataset. For the classification problem, the procedure begins by searching each variable for the cutpoint that best separates the data into two groups, each of which is internally more similar to its own members than to the members of the other group. All the possible cutpoints for maximizing the groups' internal homogeneity are evaluated. The best one is chosen, the data are thereby split into two groups, and the procedure repeats itself on each of the two new groups, making more splits and further re-examinations. At each stage of the recursion, the category diversity within groups is minimized. When no further improvement in homogeneity is possible, the algorithm stops. Recursive partitioning regression is very good at finding local, low-dimensional structure in functions that show high-dimensional global dependence. It also has a powerful graphic representation as a decision tree, the use of which is described in Section 9.2.

Once a decision rule has been generated, the rule's misclassification rates, or predictive accuracies, are estimated. A ten-fold cross-validation procedure is used to determine the best decision rule possible. The procedure works by randomly dividing the data into ten groups of equal size, and building a decision tree based on 90% of the data. The tree's misclassification rate is then assessed on the remaining 10% of the data. This is repeated for each of the ten groups, and the best decision tree is determined to be the one that produced the lowest misclassification rate. The process of cross validation thereby uses each one-tenth of the data as a test sample for the other nine tenths.

9. Results

Six hundred faults were scheduled for automatic injection into five different networks. Of these, fourteen injections failed for technical reasons, leaving 586 successful injections. From these 586 were extracted 544 feature vectors. The missing 42 vectors were not extracted due to unexpected (but reliably detectable and repairable) synchronization problems in the software architecture for the feature vector extraction process.

9.1. Detection

Overall detection coverage on any parameter was 90.6%. The greatest detection difficulty was found on network 2, which has a machine population of 150, consisting of 76 low performance systems and 71 high performance systems. Of these, three are servers. Network 3 experienced degraded detection coverage, too. This network's machine population is 40, consisting of 11 low performance systems and 14 high performance systems. Of these, 6 low performance systems and 9 high performance systems are servers. Referring to Figure 4-1, it can be seen that these two networks are the busiest of the five under test. Network 2 has the least stable traffic pattern; network 3 has a more stable mean, helping to raise the probability of correct detection, although its variance remains fairly large.

It is interesting to note the pattern of detections. The first column in Table 9-1 shows the percent detections for each of the five injected faults on network 1; total detection coverage was 93.4%. The next two columns show that the same faults were detected less well on

Roy A. Maxion and Robert T. Olszewski. Detection and Discrimination of Injected Network Faults. In *The Twenty-Third International Symposium on Fault-Tolerant Computing,* pages 198-207. IEEE Computer Society Press, 1993.

		Network				
		1	2	3	4	5
Fault	1	1.00	1.00	1.00	1.00	1.00
	2	1.00	0.88	0.79	1.00	1.00
	3	0.67	0.21	0.13	1.00	1.00
	4	1.00	1.00	1.00	1.00	1.00
	5	1.00	0.96	1.00	1.00	1.00

Table 9-1: Detection coverage (%) for all faults and networks.

networks 2 and 3. This may be because these networks carry substantially more traffic than networks 1, 4 and 5, or because the variability in traffic conditions on these networks is fairly large. Network 2 has the lowest detection coverage, and it also has the highest traffic and the highest variability (see Figure 4-1); it also has the greatest trend component (see Figure 4-3). Network 3, whose detection coverage was 78.4%, carries a high traffic load, but a fairly stable one, without substantial variability or trend. Networks 4 and 5, both quieter networks, had a 100% detection coverage, even though the variability on network 5 was relatively higher than that on network 4.

The rows in Table 9-1 show the detection coverages for each fault injected on all networks. Fault 4 (jabber), with 100% detection, seems to have been easiest to detect. Fault 3 (bad frame-check sequence) was most difficult. The relative ease and difficulty of detection has interacted in expected ways with network conditions.

9.2. Discrimination

The overall misclassification rate across all faults and all networks was 13.2%, or correct classification rate of 86.8%, as calculated from the actual correct classifications in the vector data. The cross validation classification probability matrix in Table 9-2 shows the percent of correct classifications for all of the injected faults. For example, fault 4 was classified correctly 98% of the time.

From the table it can be seen that most of the discrimination error occurred on fault 5 (broadcast). Fault 5 was misclassified as fault 1 (60-byte packet storm) 1/3 of the time. This is not surprising, however, because even a human could easily make this mistake; the two faults are parametrically very similar and easily confused[1]. Moreover, it was understood at the outset that these two fault conditions were similar. If fault 5 is excluded from the analysis, the misclassification rate goes down to 7.9%, or 92.1% successful classifications. This can be seen in the matrix in Table 9-3.

[1]Sometimes the cross-validation matrix is called a confusability matrix, because it depicts elements in a manner that demonstrates the extent to which one is confused with another.

		True Class				
Predicted Class		1	2	3	4	5
	1	0.92	0.00	0.03	0.00	0.33
	2	0.01	0.89	0.07	0.01	0.00
	3	0.00	0.07	0.86	0.00	0.00
	4	0.00	0.01	0.01	0.98	0.00
	5	0.08	0.03	0.03	0.01	0.67

Table 9-2: Cross validation classification probability matrix: all faults.

		True Class			
Predicted Class		1	2	3	4
	1	0.99	0.03	0.03	0.00
	2	0.01	0.89	0.14	0.00
	3	0.00	0.07	0.81	0.01
	4	0.00	0.01	0.02	0.99

Table 9-3: Cross validation classification probability matrix: fault #5 excluded.

Similarly, if fault 1 is excluded, the misclassification rate is 6.1%, or 93.9% correct classification. The cross validation classification matrix is shown in Table 9-4.

		True Class			
Predicted Class		2	3	4	5
	2	0.97	0.01	0.04	0.01
	3	0.00	0.93	0.09	0.00
	4	0.03	0.05	0.87	0.00
	5	0.00	0.01	0.00	0.99

Table 9-4: Cross validation classification probability matrix: fault #1 excluded.

Decision rule. The particular decision rule generated for the network fault data contains 18 clauses, and so is too long to reproduce here. Instead, Table 9-5 shows a simpler decision rule generated for the same data, but with less stringent accuracy requirements; it has only six clauses. In reading the rule, the splitting strategy of the partitioning algorithm can be seen. The overall clas-

Roy A. Maxion and Robert T. Olszewski. Detection and Discrimination of Injected Network Faults. In *The Twenty-Third International Symposium on Fault-Tolerant Computing,* pages 198-207. IEEE Computer Society Press, 1993.

sification accuracy for this rule is 81.1% as opposed to the more complex rule for the same data that discriminates at the 86.8% level. The decision rule in Table 9-5 can be interpreted as follows: first, obtain the values of the feature vector (e.g., as in Table 8-2); test these values at node 1 of the decision rule; if the value of the feature named PL0127 (code for packet lengths between 64 and 127 bytes) is greater than 5.00e-01, then the fault is of type 4, and the decision process is complete; otherwise, go to node 2 and test on the value of the feature named PL0063; this clause of the decision rule will not select a fault type, but rather will determine which clause of the decision rule to execute next; and so on. Comparing with the classification tree in Figure 9-1, if one begins at node 1 and takes the rightmost decision path at the first opportunity, the fault will be classified as type F-4, shown at the lower right of the tree. The nodes in the tree represent the clauses (nodes) in the decision rule in Table 9-5.

```
Read feature vector; then start at Node 1:

Node 1: IF PL0127  .le.  5.00e-01 THEN GOTO Node 2
        ELSE fault = type 4
Node 2: IF PL0063  .le.  5.00e-01 THEN GOTO Node 3
        ELSE GOTO Node 4
Node 3: IF DESADSU .le.  5.00e-01 THEN fault = type 3
        ELSE fault = type 2
Node 4: IF DESADSU .le.  5.00e-01 THEN fault = type 5
        ELSE GOTO Node 5
Node 5: IF DESADIN .le.  1.50e+00 THEN GOTO Node 6
        ELSE fault = type 5
Node 6: IF SOUADIN .le.  1.50e+00 THEN fault = type 1
        ELSE fault = type 5
```

Table 9-5: Decision rule for tree in Figure 9-1. PL0127 is the code for feature 16, packet length 64-127; other codes similar.

Classification tree. The simple classification tree shown in Figure 9-1 matches the decision rule shown in Table 9-5. It is used as an illustration, because it is simpler and easier to depict than the decision tree that achieves 86.8%. The tree shown has seven terminal nodes, labeled with their respective faults, while the 86.8% tree has 19 terminal nodes. Note that fault 5 is represented at three terminal nodes. The decision rule for the latter tree is considerably more complex, but also more accurate and robust. For example, the tree shown will correctly classify only 90% of true fault 4 as fault 4, while the more complex tree achieves a 98% accuracy for the same fault.

10. Discussion

The objective of the work described here was to determine whether or not a uniform characterization of relatively disparate behaviors in continuously changing environments could be used to detect and diagnose (classify) fault conditions. The uniform characterization used was a vector of anomalies, or departures from normal behavior. The method for determining whether or not anomalies had occurred was adaptive in that it accommodated for the behavioral changes of monitored networks over time.

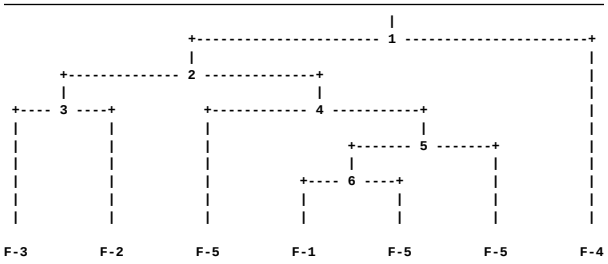


Figure 9-1: Classification tree for network faults.

The experiment was a modest success, achieving a detection accuracy of 90.6% and an overall diagnostic classification accuracy of 86.8%. This seems acceptable for a first attempt. For selected cases the accuracy was much higher. Where accuracy was impaired there appeared to be good reasons for it. For example, some networks are inherently more diverse in their behaviors than others are. Such diverse behaviors are harder to learn than are more stable behaviors. The same is true of any learning entity, even in the natural, biological world.

The analyses performed here did not address some issues that were raised during the course of the work. For example, no results are presently available that show exactly which individual faults were misclassified, although these analyses are expected in the near future. With such results it may be possible to discover a way of improving overall accuracy. One idea might be to use different, weighted, measures in a voting regime of some kind; another idea is to adjust the adaption rate for each network in accordance with that network's rate of change. (For the present study all adaption rates were identical.)

One advantage to the approach used here is that when a novel fault is detected, its feature vector can be matched against the existing set of known fault feature vectors. If the new vector is maximally similar to one of the existing feature vectors, then the new detection can be provisionally assigned to that category. Experience will then confirm whether or not the new fault is in fact in that class; if not, then a new class can be formed. Meanwhile, using the similarity test can provide a useful early means of handling novel faults.

Another idea is that adaptively generated features can be transgenerated from general functions of the original features. These new features can be used in the generation of classification rules if they are usefully discriminatory; otherwise they can be ignored if they add little discriminatory power to the generated decision rule. Efficient synthesis of transgenerated features is an open problem.

Roy A. Maxion and Robert T. Olszewski. Detection and Discrimination of Injected Network Faults. In *The Twenty-Third International Symposium on Fault-Tolerant Computing,* pages 198-207. IEEE Computer Society Press, 1993.

Although the present work was conducted in the domain of Ethernet networks, this domain was chosen more because of its operational characteristics (continuous, dynamic, noisy, nonstationary data, punctuated by injected faults) than because of its application characteristics. Electrocardiograms or seismograms or process control data could also have been used, but these were less readily available. While the work here was conducted with Ethernet networks, other networking technologies lend themselves equally well to the techniques presented here, as do many other domains.

It may seem unusual that no mention of false alarms has been made. This is because false alarms are not at issue here. All the vectors were derived from injected faults, hence no vector could have represented a no-fault, or false-alarm, condition. The only manner in which false alarms could appear in this work is in misclassification, that is, calling a fault one thing when it is actually something else. In this work there is no fault class signifying the condition in which "nothing happened." Classically, the matter of false alarms is a detection issue, not a discrimination issue.

11. Conclusions

This work has used the data from 544 usable fault injections on five networks to generate automatically a decision rule that partitions observations into correct fault classes approximately 86% of the time. Classification errors are undoubtedly attributable to noise in the monitored environments. Fault signatures consisting of vectors of performance anomalies have been generated automatically, as has the decision rule for classifying them. The procedures and algorithms used appear to be robust under a variety of environmental conditions.

12. Acknowledgements

It is a pleasure to acknowledge the contributions and comments of Paul Parker, Dave Livingston, Frank Feather and, especially, David Banks of the CMU Statistics Department.

References

[1] Avizienis, Algirdas. Architecture of Fault-Tolerant Computing Systems. In *5th International Symposium on Fault-Tolerant Computing (FTCS-5)*, pages 3-16. 1975.

[2] Bianchini, Ronald Jr. and Buskens, Richard. An Adaptive Distributed System-Level Diagnosis Algorithm and its Implementation. In *Proceedings of the 21st International Symposium on Fault-Tolerant Computing (FTCS-21)*, pages 222-229. IEEE Computer Society, Montreal, Canada, June 25-27, 1991.

[3] Bianchini, Ronald Jr.; Goodwin, Ken and Nydick, Daniel S. Practical Application and Implementation of Distributed System-Level Diagnosis Theory. In *20th International Symposium on Fault-Tolerant Computing (FTCS-20)*, pages 332-339. IEEE Computer Society, June 26-28, 1990. Newcastle upon Tyne, England.

[4] Breiman, Leo; Friedman, Jerome H.; Olshen, Richard A. and Stone, Charles J. *Classification and Regression Trees.* Wadsworth International Group, Belmont, California, 1984.

[5] Feather, Frank E. *Fault Detection in an Ethernet Network via Anomaly Detectors*. PhD thesis, Carnegie Mellon University, May, 1992. Unpublished PhD. dissertation, Department of Electrical and Computer Engineering.

[6] Iyer, Ravishankar K.; Young, Luke T. and Iyer, P. V. Krishna. Automatic Recognition of Intermittent Failures: An Experimental Study of Field Data. *IEEE Transactions on Computers* 39(4):525-537, April, 1990.

[7] Laprie, Jean-Claude. Dependable Computing and Fault Tolerance: Concepts and Terminology. In *15th International Symposium on Fault-Tolerant Computing (FTCS-15)*, pages 2-11. 1985.

[8] Lee, Inhwan; Iyer, Ravishankar K. and Tang, Dong. Error/Failure Analysis Using Event Logs from Fault Tolerant Systems. In *21st International Symposium on Fault-Tolerant Computing*, pages 10-17. IEEE Computer Society Press, Los Alamitos, California, June 25-27, 1991. Montreal, Canada.

[9] Lin, Ting-Ting Y. and Siewiorek, Daniel P. Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis. *IEEE Transactions on Reliability* 39(4):419-432, October, 1990.

[10] Little, R. J. A. and Rubin, D. B. *Statistical Analysis with Missing Data.* Wiley, New York, 1987.

[11] Maxion, Roy A. Unanticipated Behavior as a Cue for System-Level Diagnosis. In *8th International Phoenix Conference on Computers and Communications*, pages 4-8. 1989.

[12] Maxion, Roy A. Anomaly Detection for Network Diagnosis. In *20th International Symposium on Fault-Tolerant Computing (FTCS-20)*, pages 20-27. IEEE Computer Society, June 26-28, 1990. Newcastle upon Tyne, England.

[13] Maxion, Roy A. and Feather, Frank E. A Case Study of Ethernet Anomalies in a Distributed Computing Environment. *IEEE Transactions on Reliability* 39(4):433-443, October, 1990.

[14] Siewiorek, Daniel P. and Swarz, Robert S. *The Theory and Practice of Reliable System Design.* Digital Press, Bedford, MA, 1982.

[15] Thearling, Kurt H. and Iyer, Ravishankar K. Diagnostic Reasoning in Digital Systems. In *18th International Symposium on Fault-Tolerant Computing*, pages 286-291. IEEE Computer Society Press, Washington, D. C., June 27-30, 1988. Tokyo, Japan.

[16] Tsao, M.M. *Trend Analysis and Fault Prediction*. PhD thesis, Carnegie Mellon University, 1983.

Roy A. Maxion and Robert T. Olszewski. Detection and Discrimination of Injected Network Faults. In *The Twenty-Third International Symposium on Fault-Tolerant Computing,* pages 198-207. IEEE Computer Society Press, 1993.