24

A Coding Approach to Event Correlation

S. Kliger, S.Yemini
System Management Arts (SMARTS),
199 Main St., White Plains. NY 10601
kliger, yemini@smarts.com

Y. Yemini,[1]*D. Ohsie,*[2] *S. Stolfo*
450 Computer Science Building,
Columbia University, NY 10027
yemini, ohsie, sal@cs.columbia.edu

Abstract

This paper describes a novel approach to event correlation in networks based on coding techniques. Observable symptom events are viewed as a code that identifies the problems that caused them; correlation is performed by decoding the set of observed symptoms. The coding approach has been implemented in SMARTS Event Management System (SEMS), as server running under Sun Solaris 2.3. Preliminary benchmarks of the SEMS demonstrate that the coding approach provides a speedup at least two orders of magnitude over other published correlation systems. In addition, it is resilient to high rates of symptom loss and false alarms. Finally, the coding approach scales well to very large domains involving thousands of problems.

1 INTRODUCTION

Detecting and handling exceptional *events (alarms)*[3] play a central role in network management (Leinwand and Fang 1993, Stallings 1993, Lewis 1993, Dupuy et. al. 1989, Feldkuhn and Erickson 1989). Alarms indicate exceptional states or behaviors, for example, component failures, congestion, errors, or intrusion attempts. Often, a single problem will be manifested through a large number of alarms. These alarms must be correlated to pinpoint their causes so that problems can be handled effectively.

Effective correlation can lead to great improvements in the quality and costs of network operations management. For example, in a recent report on AT&T's Event Correlation Expert (ECXpert™), Nygate and Sterling (1993) report, "..labor savings at a typical US network operations center are between $500,000 and $1,000,000 a year. In addition, at least this amount is saved due to decreased network downtime." The alarm correlation problem has thus attracted increasing interest in recent years as described in a recent survey (Ohsie and Kliger 1993).

A generic alarm correlation system is depicted in Figure 1. *Monitors* typically collect managed data at network elements and detect out of tolerance conditions, generating appropriate alarms. The *correlator* uses an *event model* to analyze these alarms. The event model represents

[1] Work performed while the author was on sabbatical leave at Systems Management Arts.

[2] This author's research was supported in part by NSF grant IRI-94-13847

[3] Henceforth we use the terms *problem events* to indicate events requiring handling and *symptom events* (also *symptoms* or *alarms*) to indicate observable events. The terms *event-correlation* or *alarm-correlation* are used interchangeably to indicate a process where observed symptoms are analyzed to identify their common causes.

knowledge of various events and their causal relationships. The correlator determines the common problems that caused the observed alarms.

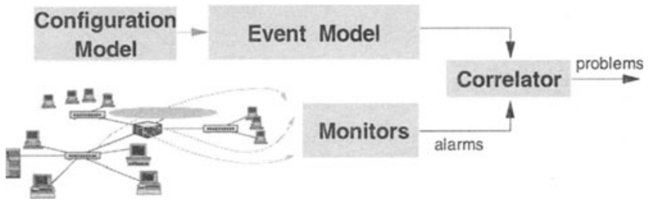

Figure 1: Generic Architecture of an Event Correlation System

An alarm correlation system must address a few technical challenges. First, it must be sufficiently *general* to handle a rapidly changing and increasing range of network systems and scenarios. Second, it must be *scalable* to large networks involving increasingly complex elements. As elements become more complex, the number of problems associated with their operations as well as the number of symptoms that they can cause increases rapidly. Furthermore, propagation of events among related elements can cause dramatic increase in the number of symptoms caused by a single problem. Finally, an alarm correlation system must be resilient to *"noise"* in the inputs to the correlator. This is because alarms may be lost or spuriously generated forming *observation noise* in the alarms input stream. The event-model may also be inconsistent with the actual network, due to insufficient or incorrect knowledge of the configuration model. These inconsistencies form *model noise* in the event model input to the correlator. An alarm correlation system must be robust with respect to both observation and model noise.

Current alarm correlation systems typically fall short of meeting the goals described above (Ohsie and Kliger 1993). Alarms are typically correlated through searches over the event model knowledge base. The complexity of the search seriously limits scalability. To control the search complexity, often the event model knowledge base is carefully designed to take advantage of specific specialized domain characteristics. This limits generality. There are no techniques to select an optimum set of symptoms to monitor or to determine whether observed symptoms provide sufficient information to determine problems. Finally, search techniques derive their computations from the data stored in the knowledge base and arriving alarms. Noise in this data can guide the search in the wrong direction. A more detailed analysis of current correlation systems is pursued in (Ohsie and Kliger 1993).

This paper describes a novel approach to correlation based on coding techniques (Kliger et. al. 1994a). The underlying idea of the coding technique is simple. Problem events are viewed as messages generated by the system and "encoded" in sets of alarms that they cause. The problem of correlation is viewed as decoding these alarms to identify the message. The coding technique proceeds in two phases. In the *codebook selection* phase, an optimal subset of alarms, the *codebook* is selected to be monitored. This codebook is selected to optimally pinpoint the problems of interest and ensure a required level of noise insensitivity. In the *decoding* phase, observed alarms are analyzed to identify the problems that caused them. The coding approach thus reduces the complexity of real-time correlation analysis through preprocessing of the event knowledge model. The codebook selection dramatically reduces the number of alarms that must

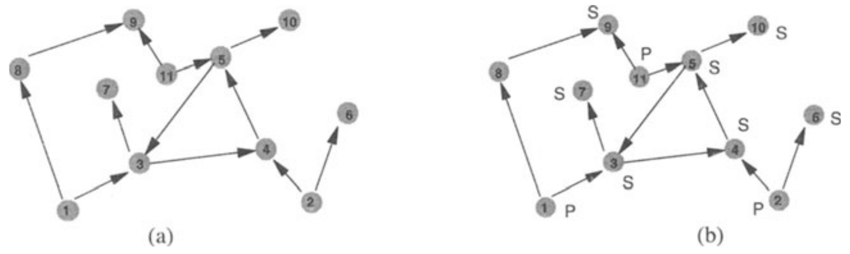

Figure 2: A Causality graph (a) and its labeling (b)

be monitored. It also establishes the relations among these alarms and their causes in a manner that reduces the complexity of the decoding phase.

In what follows we describe the mathematical basis of the coding approach (section 2), develop the technique and establish its properties (section 3), describe a commercial implementation of the coding techniques and a benchmarking of the implementation (section 4) and conclude (section 5).

2 THE MATHEMATICAL BASIS OF EVENT CORRELATION

2.1 Causality Graph Models

Correlation is concerned with analysis of causal relations among events. We use the notation e→f to denote causality of the event f by the event e. Causality is a partial order relation between events. The relation → may be described by a *causality graph* whose nodes represent events and whose directed edges represent causality. Figure 2(a) depicts a causality graph on a set of 11 events.

To proceed with correlation analysis, it is necessary to identify the nodes in the causality graph corresponding to symptoms and those corresponding to problems. A *problem* is an event that may require handling while a *symptom* (alarm) is an event that is observable. Nodes of a causality graph may be marked as problems (P) or symptoms (S) as in Figure 2(b). Note that some events may be neither problems nor symptoms (e.g., event 8) while some other events are both symptoms and problems.

The causality graph may include information that does not contribute to correlation analysis. For example, a cycle (such as events 3,4,5) represents causal equivalence. A cycle of events may thus be aggregated into a single event. Similarly, certain symptoms are not directly caused by any problem (e.g., symptoms 7,10) but only by other symptoms. They do not contribute any information about problems that is not already provided by these other symptoms that cause them. These *indirect symptoms* may be eliminated without loss of information. Henceforth, we will assume that a cauality graph has been appropriately pruned.

2.2 Modeling Causal Likelihood

The causality graphs described so far do not include a model of the likelihood (strength) of causality. The causal implication e→f can be considered as a representation of a proposition "e may-cause f." Often, richer information is available describing the likelihood of such causality. Various approaches and measures have been pursued to model such likelihood. A probabilistic model, for example, associates a conditional probability with a causal implication while fuzzy logic associates a fuzzy measure. Each of these models includes operations to compute the strength of a causal chain between two events or to combine the strength of multiple chains between two events. It is useful to have a general model of likelihood that captures these various techniques as special cases. This model must include a set of causal likelihood measures and operations to compute strength of chains and combine them. We proceed to define and demonstrate such a general model of likelihood.

Define a *semi-ring* as a partially ordered set **L** with an order $\leq$ and two operations $*$ (catenation) and + (combination) such that:

(i) $<\mathbf{L}, *>$ is a semi-group with a unit **1** (a monoid)

(ii) $<\mathbf{L}, +>$ is a commutative semi-group with a unit **0**

(iii) $\forall a,b \in \mathbf{L},\ a*b \leq a,b\ \ a,b \leq a+b$

(iv) $\forall a \in \mathrm{L},\ \mathbf{0} \leq a \leq \mathbf{1}$

A semi-ring is used to provide a measure of causality. Elements of **L** provide measures of causal strength with **1** indicating the strongest causality and **0** the weakest. The ordering of likelihood measures is used to compare relative strength of likelihood. The catenation operation is used to compute the strength of causal chains. The combination operation is used to compute the strength of multiple causal chains leading from one event to another.

We give a few examples of semi-rings used to model causal likelihood. The *deterministic model,* uses $\mathbf{L}=\mathbf{D}=\{0,1\}$ with the order $0\leq 1$. The catenation operation is the Boolean **and** $\wedge$, with the unit 1, while the combination operation is the Boolean **or** $\vee$ with the unit 0. Consider now a causality graph whose edges are all labeled with elements from **D**. An edge marked 0 represents a highly unlikely causality while an edge marked 1 represents a sure causality. For simplicity assume that all edges marked 0 have been eliminated. The semi-ring structure permits us to assign likelihood to causal chains between two events. The deterministic likelihood of a causal chain such as $1 \rightarrow 8 \rightarrow 9$ in Figure 2 is obtained by catenation (**and**) and is trivially 1. Now consider the set of causal chains between two events. The likelihood of this set is obtained by applying the combination operation to the likelihood of all causal chains in the set.

The deterministic model is a simple and commonly used likelihood model. We now introduce another semi-ring, denoted **P**, to model probabilistic causality. **P** consists of the set $[0,1]$ with an ordinary numerical order. The label q on $e \rightarrow f$ models the conditional probability of the event f when e occurs. The catenation operation is the product of probabilities while the combination operation is defined as $q_1+q_2=1-(1-q_1)(1-q_2)$.

The temporal model is denoted **T**. The elements of **T** are non-negative real numbers representing the expected duration for the respective causality to happen. For example, a label of 8.5 on $1 \rightarrow 3$ indicates that this causal implication is expected to occur within 8.5 time units (e.g., seconds). The catenation operator $*$ is addition of times (along a causal chain) while the combination operator + is the min operator on real numbers. 0 is the unit with respect to catenation, and ∞ the unit with respect to addition, where ∞ indicates that the causality is unlikely to happen (in any finite time). We use the inverted numerical order as the order on **T**, modeling "sooner" occurrence of events in time. For example, 6.3≥8.5 should be read as "6.3 happens sooner than 8.5".

Similarly, one can establish fuzzy logic models of causal likelihood or other calculus of uncertainty measures such as the Shafer-Dempster model. Furthermore, by combining various models, more complex likelihood measures may be obtained. For example, the semi-ring defined by **PxT** ascribes to a causal edge both probability and expected time of occurrence.

We are now ready to define a *causal likelihood model* as a triplet <N, **L**,ϕ> where **N** is a normal form causality graph, **L** is a semi-ring describing a likelihood model and ϕ is a mapping from the edge-set of **N** to **L** assigning a likelihood measure to each causal implication. By varying the semi-ring **L**, a spectrum of models is obtained.

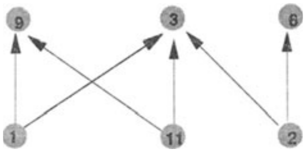

Figure 3: A Correlation Graph

2.3 The Correlation Problem

Correlation analysis is concerned with the relationships among problems and the symptoms that they may cause. Consider the *correlation relation* among problems and symptoms, defined as the closure of the relation → and denoted by ⇒. A correlation p⇒s means that problem p can cause a chain of events leading to the symptom s. This correlation relation may be represented in terms of a bipartite *correlation graph*. Figure 3 depicts the correlation graph corresponding to the causality graph of Figure 2 after pruning indirect symptoms and aggregating cycles.

For a given causal likelihood model <N,L,ϕ> one can derive a correlation graph **N*** corresponding to the causality graph **N**. Using the catenation operation one can associate a likelihood measure with every causal chain leading from a problem p to a symptom s. The likelihoods of various chains leading from p to s may be combined using the combination operator to provide a likelihood measure of the correlation p⇒s. Thus, for a given causal likelihood model <N,L,ϕ> there is a corresponding *correlation likelihood model* <N*,L,ϕ > over the correlation graph.

3 THE CODING APPROACH TO ALARM CORRELATION

3.1 Problems, Codes and Correlation

The problem of alarm correlation may be now described in terms of the correlation likelihood model. For each problem p, the correlation graph provides a vector of correlation likelihood measures associated with the various symptoms. We denote this likelihood vector as p and call it the *code* of the problem p. Codes summarize the information available about correlation among symptoms and problems. Code vectors can be best considered as points in an |S|-dimensional space associated with the set of symptoms S, which we call the *symptom space*. Alarms too may be described as *alarm vectors* in symptom space assigning likelihood measures **1** and **0** to observed and unobserved symptoms respectively. A very useful reference for coding theory and techniques is provided by [Roman 1992].

The alarm correlation problem is that of finding problems whose codes optimally match an observed alarm vector. We illustrate these considerations using the example of Figure 3. Figure 4(a) depicts a deterministic correlation likelihood model and Figure 4(b) depicts a probabilistic model. Code vectors correspond to the likelihood of the symptoms 3,6,9 in this order. They are given by $\underline{1}$=(1,0,1), $\underline{2}$=(1,1,0) and $\underline{11}$= (1,0,1) for the deterministic model and by $\underline{1}$=(0.8,0,0.3),

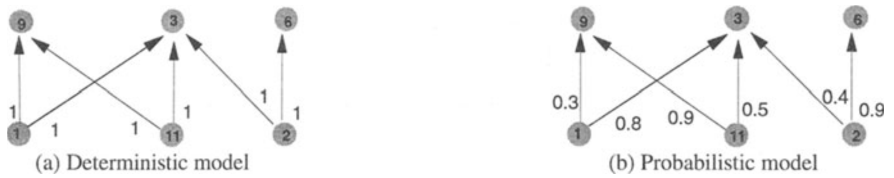

Figure 4: Correlation Likelihood Models

$\underline{2}$=(0.4, 0.9,0) and $\underline{11}$=(0.5,0,0.9) for the probabilistic model.

Suppose that alarms consisting of symptoms 3 and 9 have been observed. This may be described by an *alarm vector* $\underline{a}$=(1,0,1). In the deterministic model either $\underline{1}$ or $\underline{11}$ match the observation $\underline{a}$ and one would infer that the two alarms are correlated with either problem 1 or 11. Note that these two problems have identical codes and are indistinguishable. Similarly, an alarm vector $\underline{a}$=(1,1,0) would match the code of problem 2. How should an alarm vector $\underline{a}$=(0,1,0) be interpreted? One possibility is that this is just a spurious false alarm. Another possibility is that problem 2 occurred but the symptom 3 was lost. The choice of interpretation depends on whether loss is more likely than spurious generation of alarms. There are, of course, other more remote possibilities.

Now, suppose that spurious or lost symptoms are unlikely. The information provided by symptom 9 is redundant. If only symptoms 3 and 6 are observed the respective projections of the codes $\underline{1}$=$\underline{11}$=(1,0) and $\underline{2}$=(1,1) are sufficient to distinguish and correlate alarm vectors. Since real alarm correlation problems typically involve significant redundancy. The number of symptoms associated with a single problem may be very large. A much smaller set of symptoms can be selected to accomplish a desired level of distinction among problems. We call such a subset of symptoms a *codebook*. The complexity of correlation is a function of the number of symptoms in the codebook. An optimal codebook can thus reduce the complexity of correlation substantially.

To illustrate this consider an example of 6 problems and 20 symptoms depicted in Figure 5(a). The correlation likelihood model is compactly described in terms of a matrix. Matrix elements represent the correlation likelihood parameters of respective problem-symptom pairs.

	p_1	p_2	p_3	p_4	p_5	p_6
1	1	0	0	1	0	1
2	1	1	1	1	0	0
3	1	1	0	1	0	0
4	1	0	1	0	1	0
5	1	0	1	1	1	0
6	1	1	1	0	0	1
7	1	0	1	0	0	0
8	1	0	0	1	1	1
9	0	1	0	0	1	1
10	0	1	1	1	0	0
11	0	0	0	1	1	0
12	0	1	0	1	0	0
13	0	1	0	1	1	1
14	0	0	0	0	0	1
15	0	0	1	0	1	1
16	0	1	1	0	0	1
17	0	1	0	1	1	0
18	0	1	1	1	0	0
19	0	1	1	0	1	0
20	0	0	0	0	1	1

(a) Correlation Matrix

	p_1	p_2	p_3	p_4	p_5	p_6
1	1	0	0	1	0	1
2	1	1	1	1	0	0
4	1	0	1	0	1	0

(b) A Codebook of Radius 0.5

	p_1	p_2	p_3	p_4	p_5	p_6
1	1	0	0	1	0	1
3	1	1	0	1	0	0
4	1	0	1	0	1	0
6	1	1	1	0	0	1
9	0	1	0	0	1	1
18	0	1	1	1	0	0

(c) A Codebook of Radius 1.5

Figure 5: A deterministic correlation matrix and codebooks

Figure 5(b) depicts a codebook consisting of 3 symptoms {1,2,4}. This codebook distinguishes among all 6 problems. However, it can only guarantee distinction by a single symptom. For example, problems p_2 and p_3 are distinguished by symptom 4. A loss or a spurious generation of this symptom will result in potential decoding error. Distinction among problems is measured by the Hamming distance between their codes. The *radius* of a codebook is one half of the minimal Hamming distance among codes. When the radius is 0.5, the code provides distinction among problems but is not resilient to noise. To illustrate resiliency to noise consider the codebook of Figure 5(c) where 6 symptoms are used to produce a codebook of radius 1.5. This means that a loss or a spurious generation of any two symptoms can be detected and any single-symptom error can be corrected.

We illustrate the error-correction capabilities of the codebook of Figure 5(c). A minimal-distance decoder will decode as p_1 all alarms that contain a single-symptom perturbation of p_1. The alarm vectors {011100, 101100, 110100, 111000} will be decoded as a single symptom loss in p_1, while {111110, 111101} will be interpreted as occurrence of a spurious symptom. The total number of alarms that can be generated due to a single symptom perturbation (loss or spurious one) in the 6 problems codes + the null problem p_0=000000 is 42. Therefore, a total of 48 alarm vectors (out of possible 63) will be correctly decoded despite single-symptom observation errors. When two symptom errors occur a minimal distance decoder can detect that errors have occurred but may not decode the alarm vector uniquely.

The considerations above generalize simply to correct observation errors in k symptoms and detect 2k errors as long as k is smaller than the radius of the codebook. Consider now the problem of model errors. That is, what happens when the correlation model itself is incorrect?

For example, suppose problem p_4 in Figure 5 can actually cause symptom 6 even though the model fails to reflect this. This will cause a single symptom error with respect to the code of p_4. Symptom 6 will appear as a spurious symptom whenever p_4 occurs. In other words, *an error in* *the correlation model is entirely equivalent to an observation error.* In contrast to random observation errors, model errors would appear as persistent observation noise. This persistence may be automatically detected by analyzing correlation logs and then used to correct the correlation model.

In summary, one seeks to design minimal codebooks that accomplish a desired level of insensitivity to observation and model errors. This insensitivity to observation errors is measured by the degree to which codes are distinct. In the case of the deterministic model, distinction among codes is measured by the Hamming distance among code vectors. We will soon see that similar measures of distinction may be used to select optimum codebooks in the case of other likelihood models.

3.2 Coding and Decoding

The coding technique accomplishes significant correlation speeds. Most of the complexity of correlation computations is handled during the pre-processing of codebook selection. The decoding of alarms in real-time can be very fast. Precise complexity evaluation is beyond the scope of this paper and is left for future publications. However, even crude estimates can usefully illustrate the speed gains. The complexity of decoding is logarithmic in the number of direct decodes (alarm vectors whose errors with respect to codes are less than half the radius of the codebook). The number of direct decodes is bounded by $\delta(p,c,k)= (p+1) \sum_{m=0}^{k} \binom{c}{m}$ where p is the number of problems, c is the codebook size (number of symptoms in the codebook) and k is the number of error symptoms to be corrected ($\lceil$radius$\rceil$ - 1). The complexity of decoding is bounded by $\lambda(p,c,k)=\lg[(p+1) \sum_{m=0}^{k} \binom{c}{m}]$. For k<<p, this is of order (k+1)lgp.

In the example of Figure 5(c) p=6, c=6, k=1, the decoding complexity is $\lambda(6,6,1)=\lg[7(1+6)]=\lg 49\sim 6$ search operations. When p=100 and k=1, c may be of the order 10-30 and the complexity of decoding is of the order of 10-12 search operations. Even when $p=10^6$, decoding complexity is of a manageable order of 20(k+1) search operations. In contrast, other knowledge based approaches typically requires an exponential, or even doubly exponential, search in the total number of problems p and symptoms s (s>>c). For p=100 the search complexity may be practically infeasible. For example, Nygate and Sterling (1993) report alarm correlation speeds of ECXpert™ at 0.25 alarms per second for a model involving 10 problems.

We proceed to complete the details of codebook design and decoding for a general correlation likelihood model. The point of departure in codebook design is to define a metric of distinction among codes, generalizing the Hamming distance. This is accomplished by using a distance measure on the likelihood semi-ring **L**. We call a real function d(a,b) on **L** a *distance measure* on **L** if it is symmetric, non-negative and satisfies d(a,a)=0 and $d(a,c)\leq d(a,b)+d(b,c)$ for all $a\leq b\leq c$ in **L**. Given a distance measure d(a,b) on **L**, one can extend it to a measure of distinction among code vectors. Define the distance between two code vectors $\underline{a}=(a_1,a_2,...a_n)$ and $\underline{b}=(b_1,b_2,...b_n)$ as $\mathbf{d}(\underline{a},\underline{b})=\Sigma_{k=1,n}\, d(a_k,b_k)$. For example, in the case of the deterministic model define d(1,1)=d(0,0)=0 and d(1,0)=1 to obtain the Hamming distance.

For the probabilistic model P, a distance measure is given by the log-likelihood measure $d(a,b)=|\lg(a/b)|$ (with lg(0/0)=0 and lg(0/a)=1 for $a\neq 0$). For example, in the probabilistic model of

Figure 5(b), $\underline{1}$=(0.8,0,0.3) , $\underline{11}$=(0.5,0,0.9) and thus $\mathbf{d}(\underline{1},\underline{11})$= lg(8/5)+lg(9/3)=lg(24/5). Therefore, in the probabilistic model problems 1 and 11 are distinct, in contrast to the deterministic model. Note that the log-likelihood distance measure generalizes the Hamming distance. When all probabilities are 1 or 0 the two measures yield the same distance.

The *radius* of a set of codes P is defined as the one half of the minimal distance among pairs of codes. The radius provides worst case measure of distinction among code vectors. A codebook C is a subset of the set of symptoms S. The code space defined by C is the respective projection of code vectors in the symptom space defined by S. The *radius of the codebook*, $r_C(P)$ is the respective radius among the projections of codes. Clearly, $r_C(P) \leq r_S(P)$. Given a desired level of distinction $d \leq r_S(P)$, the *codebook design problem* is that of finding a minimal codebook C for which $d \leq r_C(P)$. Such codebook provides a guaranteed distinction of at least d among the codes of different problems.

The codebook design problem may be solved by a variety of algorithms. A pruning algorithm, for example, can start with the correlation matrix model and eliminate symptoms until an optimal codebook has been established. The algorithm may be designed independently of the specific likelihood model (semi-ring) and distance measure. This may be used to construct a correlator of great generality.

Given a codebook C, consider now an alarm vector $\underline{a}$ describing observed symptoms. The problem of decoding, as discussed above, is to find problem codes that maximally match $\underline{a}$. It is useful to utilize a correlation measure $\mu(\underline{a},\underline{p})$ for decoding that is, in general, different from the measure of distinction. We illustrate this through the example of Figure 5(a). Suppose the codebook consists of the symptoms C={3,6}. The codes are $\underline{1}$=$\underline{11}$=(1,0) and $\underline{2}$=(1,1). Now consider an alarm vector $\underline{a}$=(0,1). The Hamming distances are $\mathbf{d}(\underline{a},\underline{1})=\mathbf{d}(\underline{a},\underline{0})=1$ and $\mathbf{d}(\underline{a},\underline{2})=2$, where $\underline{0}$ =(0,0) is the null problem. Thus the Hamming distance does not distinguish between a lost symptom (correlating $\underline{a}$ with $\underline{1}$) and a spurious symptom (correlating $\underline{a}$ with $\underline{0}$).

In general, a symmetric measure does not distinguish lost from spurious symptoms. We thus permit the correlation measure to be asymmetric. A *correlation measure* on **L** consists of two non-negative functions, $\mu(\mathbf{1},a)$ and $\mu(\mathbf{0},a)$ defined for all $a \in \mathbf{L}$ such that $\mu(\mathbf{1},\mathbf{1}) = \mu(\mathbf{0},\mathbf{0}) = 0$ and if $a \leq b$, $\mu(\mathbf{1},a) \geq \mu(\mathbf{1},b)$ while $\mu(\mathbf{0},a) \leq \mu(\mathbf{0},b)$. For example, in the deterministic case define $\mu(0,1)=\alpha$ as the correlation level between an observation of no symptom when the codebook predicts its occurrence, i.e., a lost symptom. Define $\mu(1,0)=\beta$ as the correlation measure between an observation of a symptom that is not included in a code, i.e., a spurious symptom. A correlation measure on **L** may be easily extended to a correlation measure $\mu(\underline{a},\underline{p})$ between alarm vectors and code vectors. The *problem of decoding* is to find for a given alarm vector $\underline{a}$ the problem codes that minimize the correlation measure $\mu(\underline{a},\underline{p})$ -- best match problems.

In the probabilistic case let $\mu(\mathbf{1},a) = |\lg a|$ (correlation of occurrence) $\mu(\mathbf{0},a) = |\lg(1-a)|$ (correlation of non-occurrence). The correlation measure $\mu(\underline{a},\underline{p})$ is given by the logarithm of the product of probabilities assigned by $\underline{p}$ to events occurring in $\underline{a}$ and the complements of the probabilities assigned to events that do not occur in $\underline{a}$.

To illustrate the use of correlation measure in decoding consider again the codebook of Figure 5(c). The correlation measure for the deterministic model is given by $\mu(0,1)=\alpha$ (loss) and $\mu(1,0)=\beta$ (spurious). The codebook provides guaranteed error-correction for all single symptom errors.

Consider the case when two possible symptom errors occurred. For example, let the alarm vector observed be $\underline{a}$=101000. The respective values of the correlation measure for the six problems are 2α, $2\alpha+4\beta$, $2\alpha+\beta$, $2\alpha+\beta$, $\alpha+\beta$, $2\alpha+\beta$. Under all choices of α,β the two candidates decodes are p_1 (two lost symptoms) and p_5 (one lost symptom and one spurious). If $\alpha<\beta$ (loss is more likely) problem p_1 will be decoded and if spurious symptoms are more likely, p_5 will be decoded. If both observation errors are equally likely ($\alpha=\beta$) both problems will be decoded.

Decoding can be accomplished through very fast algorithms. A range of fast decoding algorithms is provided by coding theory. See (Roman 1992) for several possible algorithms with varying tradeoffs. For example, block-decoding techniques aggregate symptoms over a time window and then decode them to find minimal distance codes.

4 IMPLEMENTATION AND BENCHMARKS

The coding technique has been implemented in SMARTS Event Management System (SEMS). In this section we briefly describe the SEMS and the benchmarks used to test its performance. A full description of the benchmarks can be found in (Kliger et. al. 1994b)

SEMS is organized as an event management server and its current implementation runs on Sun Sparcstations under Solaris 2.3. The SEMS server presents interfaces allowing clients to subscribe to problem events of interest, and provides clients with notification upon the detection of problems. Typical SEMS clients include various networked systems managers. For example, a fault manager may subscribe to various fault events, a performance manager may need to handle various congestion events or excessive delay events, while a security manager may need to detect intrusion events. Clients may also include applications running under umbrella management systems such as HP OpenView, Sunnet Manager, or IBM Netview/6000.

A model of a satellite based communications network was used to benchmark the performance of the SEMS. The modelled domain includes close to 4000 managed objects involving some 9500 problems and 6000 symptoms. Random scenarios were created by selecting random subsets of the model. For example, experiments involving 100 problems proceeded by selecting a random choice of 100 problems (out of 9500) to be monitored. All symptoms irrelevant to these problems were discarded, leaving a random number of symptoms from which a codebook was selected.

The model makes two conservative assumptions *unfavorable* to codebook correlation. It assumes an *under-instrumented* system where the number of observed symptoms is much smaller

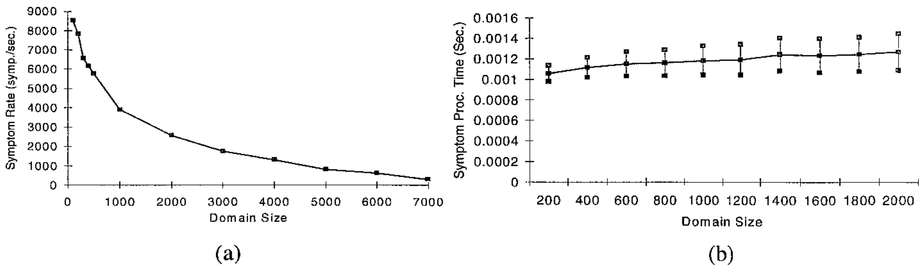

Figure 6: (a) Symptom Processing Rate (b) Symptom Processing Time with Standard Deviation

than the number of problems. Typical systems are over-instrumented. It assumes a *sparse propagation model* where only a small number of symptoms is caused by a typical problem. In a system with complex dependencies, problems can propagate very widely. Real-world situations typically monitor many more symptoms, yielding smaller codebooks, a larger reduction in the number of symptoms to monitor, and faster correlation.

The most important measure of the effectiveness of the coding approach is correlation speed. Figure 6(a) shows the effective event correlation rate measured in symptoms per second of actual elapsed time (the effective event correlation rate includes symptoms which were generated by a problem but not processed by the correlator because codebook reduction removed them from the codebook). In domains with fewer than 4000 problems, symptom processing was measured in thousands of symptoms per second. This is 2-4 orders of magnitude faster than the published figures of 0.25 events per second for ECXPERT (Nygate 1993) and 15 symptoms per second for IMPACT (Jakobson and Weissman 1993).

The fundamental measurement underlying the curve of Figure 6(a) is the elapsed time for processing symptoms. Figure 6(b) depicts these time measurements and the intervals defined by the standard deviation of the measurements. The figures shows that the average speed measures provide a fairly accurate estimate of the actual correlation rates.

Another important aspect of the coding approach its resilience to symptom loss. Figure 7(a) shows the correlation error rates when the probability of symptom loss ranges up to 20%. Even substantial loss or spurious symptoms cause only minimal error probability, falling under 5% when the codebook radius exceeds 1.5.

Our final measure of codebook performance is what reduction is accomplished in the number of symptoms that must be monitored, compared with the total number of relevant symptoms available. The compression factor represents the ratio of the two numbers. This compression is an important feature of the coding approach as it reduces the amount of monitoring and real-time processing of events needed. Figure 7(b) depicts the behavior of the compression factor as the domain size grows. The figure shows that substantial compression is achieved by the codebook.

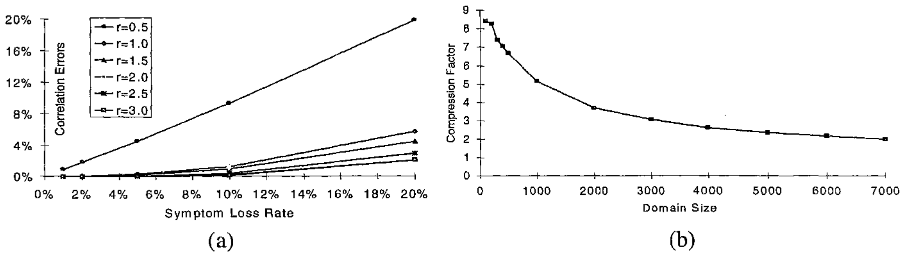

Figure 7: (a) Correlation Error Rate (b) Codebook Compression

5 CONCLUSIONS

This paper provides an overview of the coding approach to event correlation and its mathematical foundations. The coding approach accomplishes the three goals described in the introduction: generality, scalability and resilience to noise. Generality is accomplished through the use of an abstract mathematical formulation of the event correlation process. Scalability is accomplished through a substantial reduction in real time correlation processing due to optimizing symptom sets and fast decoding mechanisms. The complex searches through causality models are performed during the pre-processing phase of codebook design. Resilience to noise is accomplished by selecting codebook symptoms to provide a desired level of guaranteed noise insensitivity.

The coding approach has been implemented in SMARTS Event Management System. The current implementation runs as a server under Sun Solaris 2.3. Preliminary benchmarks confirm the advantages promised by the theoretical analysis.

6 REFERENCES

Dupuy, A., Schwartz, J., Yemini, Y., Barzilai, G. and Cahana, A. (1989) Network Fault Management: A User's View, in *Proc. IFIP Symposium on Integrated Network Management*, North Holland.

Feldkuhn, L. and Erickson, J. (1989) Event Management as a Common Functional Area of Open Systems Management, in *Proc. IFIP Symposium on Integrated Network Management*, North Holland.

Jakobson, G., Weissman, M. (1993) Alarm Correlation, *IEEE Network*, Vol. 7, No. 6.

Kliger, S., Yemini, Y. and Yemini, S. (1994a) Apparatus and Method for Event Correlation and Problem Reporting, Patent Application.

Kilger, S., Ohsie, D., Yemini, Y., Hwang W. (1994b) Decs Performance Benchmarks Summary, *SMARTS Technical Report.*

Leinwand, A., Fang, K. (1993) Network Management : A Practical Perspective Addison Wesley.

Lewis, L. (1993) A Case Base Reasoning Approach to The Resolution of Faults in Communications Networks, in Proceedings *Third International Symposium on Intégrated Network Management.*

Nygate, Yossi and Sterling, Leon (1993) ASPEN - Designing Complex Knowledge Based Systems in *Proceedings of the 10th Israeli Symposium on Artificial Intelligence, Computer Vision, and Neural Networks,* pp. 51-60.

Ohsie, D. and S. Kliger (1993) Network Event Management Survey, *SMARTS Technical Report.*

Roman, Steve (1992) *Coding and Information Theory,* Springer Verlag.

Stallings, W. (1993) *SNMP, SNMPv2, and CMIP The Practical Guide to Network-Management Standards,* Addison Wesley.

Yemini, Y., Dupuy, A., Kliger, S., Yemini, S. (1993) Semantic Modeling of Managed Information in *Second IEEE Workshop on Network Management and Control,* Tarrytown, NY.

7 BIOGRAPHY

Professor Yechiam Yemini is the director of the Distributed Computing and Communications Lab at Columbia University and a co-founder of SMARTS. His interests include broad areas distributed networked systems technologies; he has published over 100 articles and edited 3 books in these areas. **Dr. Shaula Alexander Yemini** is president and co-founder SMARTS. Her past work includes the design of the Hermes Distributed Programming Language, the Concert high level language system and the co-invention (with Rob Strom) of Optimistic Recovery, a technique for transparent fault tolerance in distributed systems. **Dr. Shmuel Kliger** leads the development of SEMS at SMARTS. His research experience includes designing and implementing distributed concurrent logic programming languages and environments. **Professor Salvatore Stolfo** heads the Parallel and Distributed Intelligent Sytems Laboratory at Columbia University, where he led the development of the PARADISER parallel and distributed database rule processing system. **David Ohsie** is a Phd. candidate at Columbia University, where he is currently pursuing his thesis research in causal analysis.