Self-Aware Services: Using Bayesian Networks for Detecting Anomalies in Internet-based Services

Alexandre Bronstein, Joydip Das, Marsha Duro, Rich Friedrich,
Gary Kleyner, Martin Mueller, Sharad Singhal, Ira Cohen[1]
Internet Storage and Systems Laboratory
HP Laboratories Palo Alto

HPL-2001-23 (R.1)
October 15th , 2001*

E-mail: {alex_bronstein, joydip_das,marsha_duro, rich_friedrich, gary_kleyner,
martin_mueller, sharad_singhal}@hp.com, iracohen@ifp.uiuc.edu

service management, anomaly detection, Bayesian networks, on-line learning, fault and performance management

We propose a general architecture and implementation for the autonomous assessment of health of arbitrary service elements, as a necessary prerequisite to self-control. We describe a health engine, the central component of our proposed 'Self-Awareness and Control' architecture. The health engine combines domain independent statistical analysis and probabilistic reasoning technology (Bayesian networks) with domain dependent measurement collection and evaluation methods. The resultant probabilistic assessment enables open, non-hierarchical communications about service element health. We demonstrate the validity of our approach using HP's corporate email service and detecting email anomalies: mail loops and a virus attack. We also present the results of applying on-line machine learning to this architecture and quantify the benefits of the Bayesian network layer.

---

* Internal Access Date Only Approved for External Publication

[1] Beckman Institute, 405 N. Mathews Ave., Urbana, IL 61801

Self-Aware Services: Using Bayesian Networks for Detecting Anomalies in Internet-based Services

A. Bronstein, J. Das, M. Duro, R. Friedrich,
G. Kleyner, M. Mueller, S. Singhal
Hewlett-Packard Company
1501 Page Mill Road
Palo Alto, CA 94304
{alex_bronstein, joydip_das, marsha_duro,
rich_friedrich, gary_kleyner,
martin_mueller, sharad_singhal}@hp.com

I. Cohen
Beckman Institute
405 N. Mathews Ave.
Urbana, IL 61801
iracohen@ifp.uiuc.edu

Abstract

We propose a general architecture and implementation for the autonomous assessment of health of arbitrary service elements, as a necessary prerequisite to self-control. We describe a health engine, the central component of our proposed 'Self-Awareness and Control' architecture. The health engine combines domain independent statistical analysis and probabilistic reasoning technology (Bayesian networks) with domain dependent measurement collection and evaluation methods. The resultant probabilistic assessment enables open, non-hierarchical communications about service element health. We demonstrate the validity of our approach using HP's corporate email service and detecting email anomalies: mail loops and a virus attack. We also present the results of applying on-line machine learning to this architecture, and quantify the benefits of the Bayesian network layer.

Keywords

Service management, anomaly detection, Bayesian networks, on-line learning, fault and performance management.

1. Introduction

Managing complex hardware and software systems has always been a difficult task. The Internet and the proliferation of web-based services have increased the importance of this task, while aggravating the problem in at least four ways:

- Internet speed software development and release means less reliable and more frequently updated software.

- Multi-tier and distributed software architectures increase the complexity of the environment and obscure causes of both functional and performance problems.
- Internet style service construction implies more dynamic dependencies among the distributed software elements of the overall services making it difficult to construct and maintain accurate system models.
- Internet scale deployments increase the number of service elements under a particular administrator's responsibility.

Currently, the paradigm for detecting problems in computing environments is to monitor many hardware, software and system operational variables across time and note the occurrence of abnormal events. The information is typically monitored by small sets of network, system and application administrators who assess, for each service element, whether that element is 'healthy' or not. In this context, the assessment of health is a determination of whether the current observed behavior is consistent with expectations. These expectations may be based on models of correct behavior or on observations over time and the patterns within those observations. The assessment of health is key to troubleshooting problems and detecting faults and failures before they propagate to the users of the system.

As the numbers of service elements and complexity of service environments have grown, so has the amount of management information, increasing the burden on IT staff. Advances toward more efficient and accurate problem detection have included:

- Applying reasoning techniques to the monitoring information to help the administrator answer the question 'is this service element healthy?' Simple examples include the commercially pervasive use of fixed or statistical thresholds and the classification of alarm levels. Commercial applications of more sophisticated reasoning technology, for example, neural networks [1], are also beginning to emerge.
- Applying reasoning techniques to the abnormal events information to help answer the question 'which service element is causing the problem?' Alarm or event correlation systems [2] are now found in many commercial products [3] [4] [5]. Rule-based or model-based expert systems associate individual abnormal events with service elements, and group events to focus on the likely causal element(s), thereby reducing the set of events that need to be presented to the administrator.

Sophisticated anomaly detection technologies are now seen in the network management and hardware management spaces. Anomaly detection in software, however, has made less progress. The advent of complex, distributed and federated services has brought a new class of challenges.

In the research community, attempts have been made to extend existing management models to include service elements as first-class objects, such as [6], [7] and [8]. In the probabilistic reasoning arena [9], Hood [10] [11] [12] has used a Bayesian network over threshold violations in measurements in a computer network to give a probabilistic assessment of the existence of a fault in the network. Our work builds on these approaches, by generalizing Hood's Bayesian network application idea to arbitrary service elements.

The rest of this paper[1] is organized as follows: in section 2, we present the model and architecture for self-aware services, and its properties. In section 3, we describe an instance of that architecture, customized to email services and detection of mail loops and virus intrusions. In section 4, we explain and present our efforts to apply machine learning to improve the accuracy. And in section 5, we analyze the advantages of the probabilistic reasoning layer in the SAC architecture. Finally we summarize our contributions and discuss directions for future work.

2. Self-aware services

2.1. Motivation: self-awareness of health as a management paradigm

A complex service can be viewed as a set of interdependent service elements or objects. Managing such a service, from the point of view of detecting anomalies in its functioning and locating the responsible sub-service elements, is a difficult task. The motivation underlying our work is that such tasks become easier when service elements are aware of their own health. In the ideal case, all service elements, at all levels of abstraction, are able to accurate assess and efficiently communicate their health status, at all times. Anomaly detection becomes trivial, and fault localization becomes simpler.

This vision leads us to propose that all service elements be equipped with machinery, logically local to each element, which accurately and sensitively evaluates the element's health. The resulting distributed health assessment architecture can then begin to address the scalability issues brought on by the Internet.

Basing the health evaluation on statistical deviations from past history obviates the need for precise models of the service element's behavior. Statistical evaluators, by using time dependent weighting, can adapt to dynamic changes in the service elements. This model independence and dynamic adaptability are advantageous in the context of complex and dynamic architectures, as well as high software churn.

Allowing model-based health evaluations within the same architecture gives us the potential for higher accuracy, when those models are available.

---

[1] This paper is a superset of the one presented at IM-2001 [32]: the experimental demonstration in section 3 contains a more recent customization with better detection results, and section 4 and 5 are new.

In addition, we believe that much of the machinery needed to compute and communicate service element health can be made largely independent of the specific element and its semantic domain. This is the basis for proposing a general architecture, whose instances are customized to a wide variety of service elements.

We define 'self-awareness' as the ability of an element to autonomously detect deviations in its behavior that are meaningful. We define 'self-control' as the ability of the element to respond to this information in a manner that appropriately changes its behavior. Accurate self-awareness is prerequisite to self-control, and in this paper, we will focus on the self-awareness aspect of the overall vision.

2.2. Health engine: sensing, evaluating and Bayesian reasoning

The computation of a single, accurate, assessment of health is key to the expected benefits of this work. We therefore explain the logical model of the health engine in detail below.

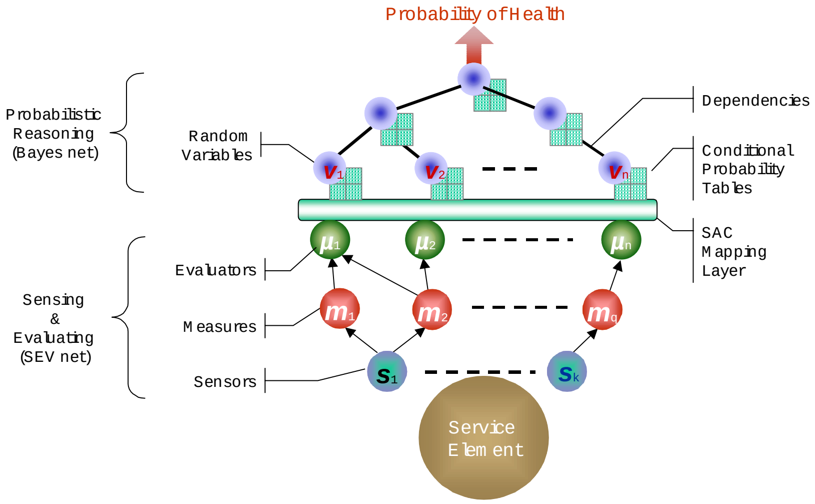


Figure 1: Health Engine Model

The health engine achieves its result by first logically wrapping 'sensors' around a particular service element. Those sensors generate measurements over time, which are stored in the 'measures'. We use the terms, sensor and measure, in a broad sense, intended to denote any information capture about the behavior of the underlying service element. Sensors may or may not require cooperation from the underlying element. Examples of cooperation-independent sensors are operating system tools that measure CPU usage, memory consumption, or I/O rates for a particular process. Alternatively, calls to management APIs of a service element (if they exist) are cooperation-dependent sensors in our terminology. Another useful and commonly available sensor is an application log parser/analyzer. For abstract service elements that do not correspond to a single process, active tests by pseudo-clients can be constructed as sensors in our framework. Where warranted, sensors that intercept the request/response flow of the underlying element also fit within the architecture.

The second layer in the health engine accumulates the information provided by the sensors in the measures. The architecture allows for information of any type to be stored there. The amount of past data to be kept is measure specific. A given sensor can contribute data to one or several measures; we denote that information flow with arrows going from the sensors to the measures in Figure 1.

The next layer is the 'evaluators'. These are functions that process one or more of the measures, to yield an opinion of the service element's health. The opinion can be expressed in binary, tertiary, or any N-ary form. The evaluators have access to a statistical library and a time-selection construct, so that 'within 2 standard deviations of the mean value in the last 30 days' can be expressed easily. The evaluators are not limited, however, and are free to exploit domain constraints either in the data extraction phase or in their statistical computation.

The top layer attempts to subtly combine the individual opinions expressed by the evaluators into a single assessment, using knowledge about the accuracy of the evaluators in different circumstances. The intuition here is that reasoning about the evaluators' opinions yields more accurate conclusions than simply 'ORing' them, as is commonly done. In our first implementation, we chose Bayesian networks in the top layer for several reasons. Bayesian networks are a proven technology in the field of diagnostics [13] [14] and are capable of leveraging prior expert opinions with learned information from data [15].

The individual health evaluations are entered as evidence (in the Bayesian reasoning sense) to the Bayesian inference engine. The customized Bayesian networks are specified to have one leaf node ('random variable') for each evaluator. The conditional probability tables encode how much weight to attach to the opinion of each evaluator. The Bayesian inference engine always functions on partial evidence, so an evaluator unable to give an opinion for any reason (e.g. lack of sensor data) does not require special treatment. Whenever an assessment of the overall health is needed, the current evaluations are entered as evidence, and the inference engine computes the resulting probability for the top node. The probability is the single assessment of health at that instant.

2.3. Expanding the semantic range of health communications

Management systems typically communicate information about element health via lists of name-value pairs (e.g. MIB readings) or notifications of abnormal events (e.g. SNMP traps). Those information types are understood only by a management overseer or monitor. By providing reduced and universal information about health (a real number between 0 and 1), and communicating via broadly accessible mechanisms (XML + HTTP), we enlarge the pool of possible applications that can communicate about a service element's health to:

- a variety of management applications beyond the top monitor;
- a peer, such as a potential customer or supplier of that service element;
- the service element itself! (Hence the name 'Self-Aware Services'…).

A frequent issue in distributed management systems is the need to standardize the language used for communicating information between service components and the management system. In our architecture, the basic information about health has been reduced to a single number and can thus be communicated using any existing language that allows tagged variables.

Although we believe that there is value in reifying the concept of element health and reducing it to a single number, we recognize that there are drawbacks. To address some of these, the architecture places no restrictions on additional information that could be communicated. For example, in the implementation we will describe later, we defined in our XML an additional 'Details' tag that allows for the inclusion of CIM-XML reports or dumps from evaluators and sensors. Further, we have anticipated the possibility of different definitions of health by not restricting the number of Bayesian networks that can be used simultaneously. This is useful in cases where applications belong to significantly different semantic domains and need to communicate different notions of health.

2.4. Self-awareness architecture

Our proposed self-awareness architecture therefore rests on three ideas:

- We generalize the measurements and thresholds concept widely found in commercial system management technology to a collection of sensors for measurement capture, measures for relevant measurements accumulation, and evaluators for the generation of opinion about the service element health, based on statistical or absolute tests.
- We combine the multiple opinions about the element health into a single, probabilistic assessment, using probabilistic reasoning technology.
- We communicate using standard web technologies. To configure the engine and communicate the health information to other service components and/or management systems, we use open XML-based descriptions transported over HTTP.

These fundamental components of the self-awareness architecture are depicted below, in Figure 2.

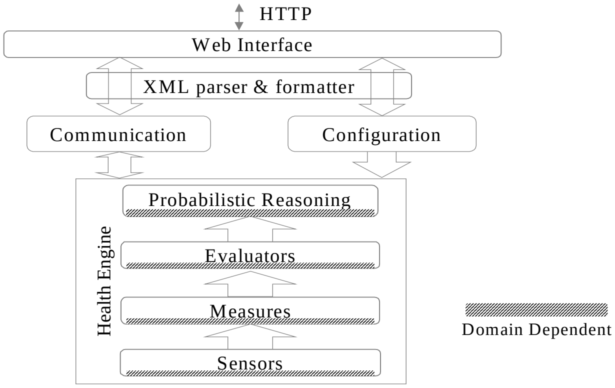


Figure 2: Health Awareness Architecture

The resulting health assessment can be fed into existing management systems to assist administrators, or into higher-level fault localization systems, such as event correlation systems. It can also be passed to a local control module for the service element itself, thereby yielding the ‘Self-Awareness and Control’ paradigm.

While largely domain independent, in implementation, the machinery is tailored to a domain by configuring a Bayesian network, and a set of associated Java classes for the element specific parts of the sensors, measures and evaluators.

2.5. The generality of the technology

The SAC architecture and machinery are general: the overall communication mechanism and SAC framework are element independent. The Bayesian network engine, and many of the mechanisms for accumulating and processing the measurements are element independent.

Semantically, there are no restrictions on the kind of service element to which the SAC concept can be applied. The application of this architecture to a service element involves customizing the sensors, picking meaningful measures, and defining appropriate evaluators for the health question at hand. Further, the parameters of the Bayesian network, that is, the conditional probability tables must be specified, reflecting any prior knowledge about the behavior of the service element.

The practical granularity of the service elements that can be made self-aware is constrained only by the resource footprint of the implementation and the resource budget of the domain.

3. Experimental validation

3.1. Virus and mail loops detection using SAC

To test the validity of the SAC approach, we have implemented this architecture a number of times, each time adding more capability. We used HP Chai [16] [17] in our first prototype as the front end web server because it provided us with a lightweight web server which is designed to allow back-end servlet applications to be integrated easily. Any other web server and servlet facilities could have been used. For the Bayesian engine, we wanted a lightweight and memory efficient engine and used Professor Fabio Cozman's EBayes [18], intended for embedded environments. The Bayesian networks were described in an XML dialect: XBN [19], slightly extended to include the linkage between the Bayesian network nodes and our evaluator class names. We used an XML parser available from IBM [12].

Abstract Java classes provide a large part of the functionality needed for building sensors and evaluators. To customize a SAC to a service element, small concrete Java classes that gather and rate relevant information are defined and written. Those classes can leverage all the functionality provided by the abstract parent classes, as well as the statistics and time selection libraries. In the experiment described below, we modified 7 Java classes, out of the more than 70 we have coded into the framework. We also coded a log file parser, specific to this experiment.

Most recently, we applied the SAC concept to the detection of a set of email anomalies on a corporate mail system. In this experiment, we attempted to detect a virus infection with no knowledge of the specific attributes of the virus. We also attempted to detect mail loops, as these also negatively impact mail systems, principally by increasing traffic and thereby using system resources. Mail loops often go undetected by email monitors unless the load becomes excessive or the administrator catches the traffic by chance.

The source of information, the sensor in SAC terminology, was the mail (postfix) log file from a corporate firewall that handles messages going in and out of the HP domain. This firewall does not see any internal email traffic. Virus detection at this level is more difficult because the intensity of the virus-induced traffic is lower at this corporate boundary than in the internal mail system. Most recent email viruses use the user's address book to send themselves out, and most address books in a company contain principally internal email addresses.

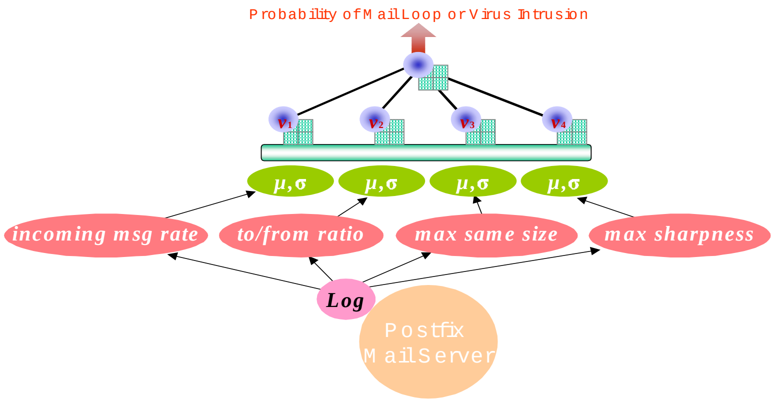


Figure 3: Structure of the Virus/Mail Loop detector

Figure 3 shows the structure of the system, including the structure of the simple Bayesian network. The measures and evaluators were chosen based on conversations with an experienced member of the corporate email support organization. His input was also used to set the parameters of the conditional probability tables, reflecting how much the particular evaluator should be trusted to indicate the presence of the targeted mail anomalies.

3.2. The sensors and measures

The mail logs record every action taken by the mail system: when a message is received, from whom and to whom, the message size, when there is an attempt to deliver a message, and the status of the attempt ('sent', 'deferred' or 'bounced'). Note that for the following definitions, 'incoming' and 'outgoing' are defined relative to the firewall mail server, not the HP domain; that is messages received by the mail server are counted as incoming whether they originate inside or outside the corporate boundary. From the logs, four measures were computed:

- The rate of received messages. The rate was defined to be the number of incoming messages observed in the measurement interval, divided by the interval size, 10 minutes in all of these experiments.
- The ratio between incoming and outgoing messages during the measurement interval. The measure was defined to be the absolute value of the difference between this ratio and 1. Note that in this email implementation, a message with multiple recipient addresses results in a single incoming message at the mail server, but multiple outgoing messages. Viruses and mail loops emails

usually have one recipient per message. In an outbreak, the ratio comes closer to 1 than during normal operation.

- The magnitude of the peak bin in the distribution of incoming message sizes. The frequency distribution of message sizes was accumulated and the bin contents normalized by the total number of messages observed in the interval. When a virus outbreak or email loop occurs, there is a spike around a particular message size. This has been observed in the recent virus outbreaks. This measure picks up mass emails as well, but the previous measure (incoming/outgoing ratio) offsets this drawback.
- The “sharpness” of the peak of the distribution of incoming message sizes. Using the bin with the maximum number of messages found in the previous measure, the sharpness is computed as the ratio between the size of that bin and the average size of its relevant neighbors. By relevant neighbor, we mean the 4 neighboring non-empty bins, excluding the two nearest neighbors on both sides of the maximum bin, to accommodate the known jitter in message size due to header information.

For this experiment, the subset of possible measures chosen was small and not specific, by intent. Many current virus detection systems rely on text strings being present in the virus (e.g. ‘Life stages’ in the subject line). New viruses can bypass these detection systems simply by changing text strings. String based solutions are sometime used for spam filtering and could also be used for mail loop detection. Our implementation did not depend on any specific pattern in the message itself, but on the effect the virus or mail loop has on the mail system, namely the change in traffic characteristics that occurs when such an anomaly is present. This allows us to detect the problem in a more attack independent manner.

3.3. The evaluators

The SAC architecture neither specifies nor restricts the type of evaluation applied to the measures. In this experiment, we used simple statistical evaluators, namely the mean (μ) and standard deviation (σ), computed over the last 2 hours, to determine whether the measures indicated a problem. We chose these evaluators with no implied assertions about the underlying statistical characteristics of the measures. In addition, we filtered the values used in computing the running μ and σ by excluding outliers, defined as beyond +/- 3.5 σ from μ. This was done to better track the normal behavior of the system and to reduce the likelihood that the evaluators quickly adapt to pathological values.

The values of the evaluators for the incoming message rate, the message size peak and sharpness were determined as follows:

$$\text{Evaluator} = \begin{cases} \text{Good} & \text{if present Measure} \le \mu + \sigma \\ \text{Medium} & \text{if } \mu + \sigma < \text{present Measure} \le \mu + 2\sigma \\ \text{Bad} & \text{if present Measure} > \mu + 2\sigma \\ \text{Don't Know} & \text{if no past measures were available to compute } \mu \text{ and } \sigma \end{cases}$$

The evaluator for the ratio of incoming and outgoing messages was determined by:

$$\text{Evaluator} = \begin{cases} \text{Good} & \text{if present Measure} \ge \max(\mu - \sigma, \frac{\mu}{2}) \\ \text{Bad} & \text{otherwise} \\ \text{Don't Know} & \text{if no past measures were available} \end{cases}$$

The above illustrates the ability of the Bayesian networks to handle ‘Don’t Know’ as an input from any evaluator.

3.4. The experiment

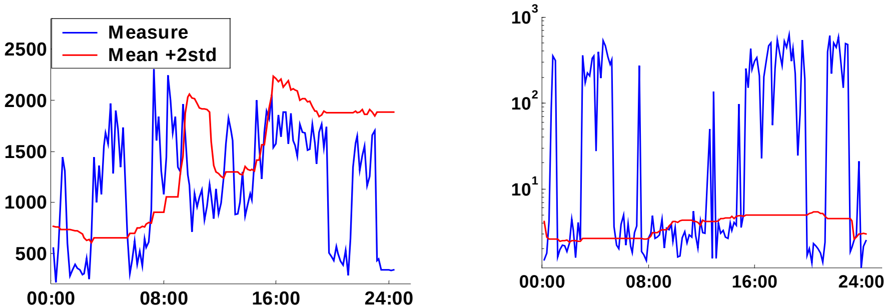
For this feasibility experiment, 17 days worth of mail logs from one of the firewall systems were analyzed. During this period, one notably virulent attack occurred (‘Life stages’ virus) and affected the system for approximately half a day before defenses were installed. Over 200 of the more than 2400 measurement intervals were impacted by significant mail loops, as established by an automated counting script described below. This data sample was deemed a good test bed for the SAC since it had enough adverse events to enable a meaningful statistical evaluation. The experiment we describe was done off-line to allow a detailed statistical analysis of the output of the system. This was an implementation choice, consistent with the research objective, not a limitation of the architecture.

On the other hand, since there was only one virus attack in our data, we could not perform a statistically meaningful validation. Without the message header or content information, it was impossible to reconstruct the beginning or end of the attack from the server log. Corporate-wide communications were sent late morning and defenses were installed across the corporation in the late afternoon of the affected day. The firewall SAC gave a sharp drop in health probability at the 11:01 AM and 12:01 PM intervals but did not indicate problems at other times during the possible virus infestation period. Because of the impracticality of doing statistical validation on the virus events, the rest of this analysis focuses on the mail loop events and detection.

3.5. Results

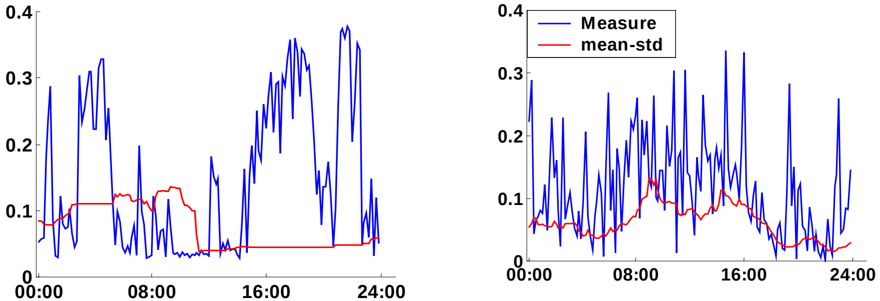
The effects of combining the outputs of several evaluators with a Bayesian network are illustrated in Figures 4 and 5. The individual evaluator graphs in Figure 4 show both random as well as systematic variability in the individual measures. Each of the four sub-figures plots one of the four defined measures, as a function of time for one 24-hour period in the 17-day experiment. The variability of these data illustrates the impracticality of using static thresholds as alarms.

Superimposed on these sub-figures are plots of the outermost decision boundary where the statistical evaluators registered ‘bad’. In sub-figures (a)-(c), corresponding to the incoming message rate, peak message sharpness and size measures, the dashed lines correspond to the recently time-averaged means plus two standard deviations. For the fourth measure, the ratio of incoming to outgoing messages, mean minus one standard deviation is shown.



(a) Incoming message rate

(b) Sharpness evaluator (logarithmic scale)



(c) Peak of message size distribution

(d) Ratio of incoming and outgoing messages

Figure 4: Plot of the measures for all 10 minutes periods of one particular day with the decision boundaries of the evaluators

In Figure 5, we display the output of the Bayesian network, corresponding to the probability of the email anomaly over the same 24-hour measurement period. The apparent improvements in signal differentiation and noise attenuation were validated and substantiated by statistical analyses using standard detection evaluation techniques that we describe next.

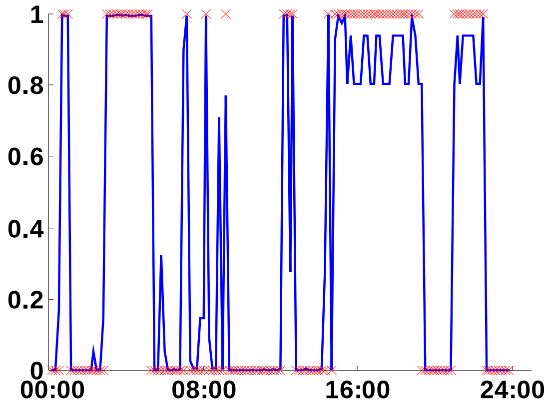


Figure 5: Probability of mail loop, computed over the same 24-hour period as Figure 4, ‘x’ mark the ground truth for the interval (1-Loop present, 0- Loop absent)

To validate the output of the Bayesian network, we used a detection evaluation method called the Receiver Operating Characteristic (ROC, extolled in [20]). The ROC is a graph showing the probability of accurate detection of an event versus the probability of a false alarm, using a moving decision threshold. In our case, the decision threshold is applied against the output of the SAC. The upper left point (0,1) denotes the ideal of complete detection with no false alarm. In general, the closer the curve comes to that point, the better.

Another useful evaluation of the detector plots the probabilities of error as a function of the decision threshold. The range of thresholds that yield optimal performance is indicative of robustness.

The computation of the ROC and error probability curves requires that we know the ground truth in the experiment, that is, that we have a way of validating if a given detector output is correct or not. In any detection experiment using real-life data, the determination of the ground truth is a critical but often difficult task, which affects the reported performance. For determining the ground truth for the loops, we used a script that reconstructed the mail traffic from the log, looking in detail at the addresses of the senders and recipients. We filtered out all messages with more than one recipient and then counted all back-and-forth message pairs between unique sender/recipient pairs. We counted the mail loop traffic in each 10-minute interval and labeled as anomalous any interval with loop traffic in excess of 10% of the peak throughput of the server.

The performance results are shown in Figures 6 and 7. The ROC curve comes very close to the ideal point mentioned above. The probability of error curves show a large range [0.35, 0.7] of decision thresholds at which the detector performs well, indicating good robustness.

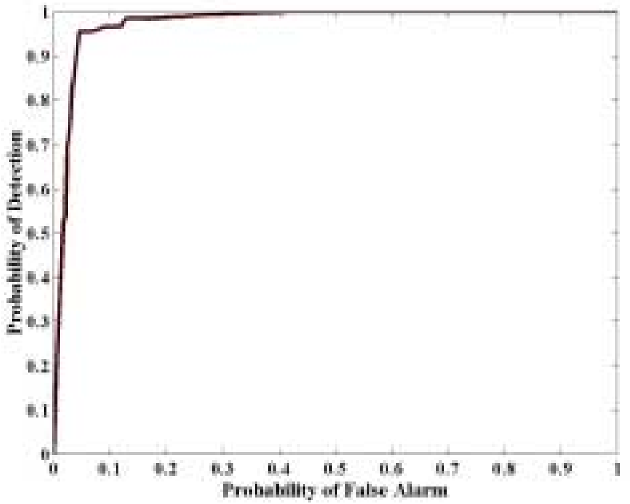


Figure 6: ROC of the email loop experiment

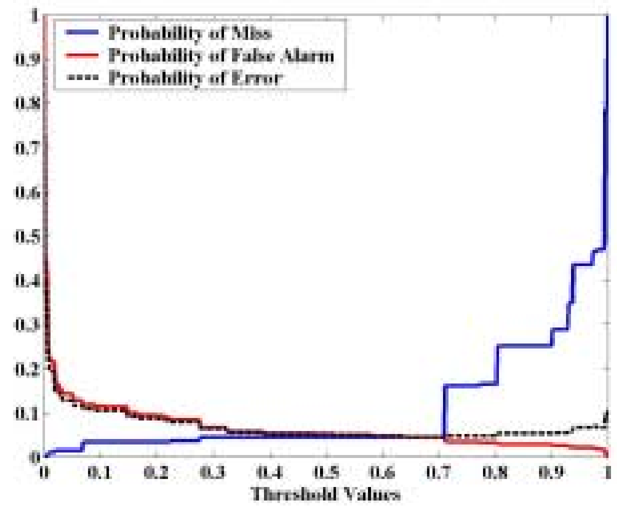


Figure 7: Probabilities of error as a function of the decision threshold

4. Application of machine learning

4.1. Opportunity and necessity

In the work described so far, the conditional probability tables (CPT's) of the Bayesian network were set using human expert knowledge. The opinions of a domain expert were used to estimate how much a particular evaluator, giving a particular answer, should be trusted to indicate the presence of the targeted email anomalies. This is a significant cost of deploying the SAC architecture to any domain. Using machine learning techniques theoretically offers the opportunity to both improve over the initial human estimates, and possibly, to eliminate the need for human estimates altogether.

In addition, there are circumstances where the characteristics of the underlying service element change (e.g. a version upgrade, or a reconfiguration). To remain accurate then, the Bayesian network model needs to adapt to the underlying dynamic reality. This calls for the use of machine learning techniques that can take advantage of on-going observations, and feedback (i.e. labeled data) when available, to adjust the CPT's accordingly. There is a more subtle form of dynamism that we encounter when the health of the underlying element in a particular interval is not independent from the health at the previous interval. An adaptive learning algorithm can change the CPT's to take advantage of such time linkage, and thus alleviate the need for a more complex non-stationary model, such as dynamic Bayesian networks [21].

Bayesian network learning is a rich field, surveyed in [15], [22], and [23]. In order to provide the adaptive capability just mentioned, and also to avoid the requirement for a batch of training data, we narrowed our focus to on-line learning, i.e. techniques that use data coming along as the system operates (e.g. [24], [25], [26]). For a couple of different reasons (adapting quickly to changes on one hand, and occasionally capitalizing on a few records of human feedback on the other hand) we needed a technique that could take advantage of a small amount of labeled records. We therefore focused on Voting EM [27], [28], an algorithm that was developed for on-line learning of Bayesian network CPT's in the context of scarce labeled training data. In the next section we present the results of that exploration.

4.2. Experimental results using Voting EM learning

For this experiment, we used the same 17 days of data described in section 3, which contained about 200 intervals with mail loop present. The learning was done incrementally, using the adaptive Voting EM algorithm, in the following manner: for every time interval, the current Bayesian network (with its current CPT's) evaluates the record and gives its answer. Then, the known ground truth classification for that interval is given to the system as a labeled training record, which then alters slightly the CPT's, thereby yielding the Bayesian network which will be used at the next time interval. The resulting vector of output probabilities was then used to generate the ROC and error probability curves in exactly the same manner as in section 3.

Structuring the learning and testing as we described guarantees that the system never 'sees' the answer for any interval before returning its own evaluation/classification first. It is the analog for on-line algorithms of splitting the training and testing data for batch algorithms. The goal is to evaluate the system as fairly as possible, and to avoid over fitting any particular data set.

The performance results are shown in Figures 8 and 9. The ROC curves of both the experiments with and without learning are shown in Figure 8. The ROC's show that on-line learning yields a significant improvement in performance. The probability of error curves show further improvement in robustness (as defined in section 3.5) in comparison to the curves shown in Figure 7.

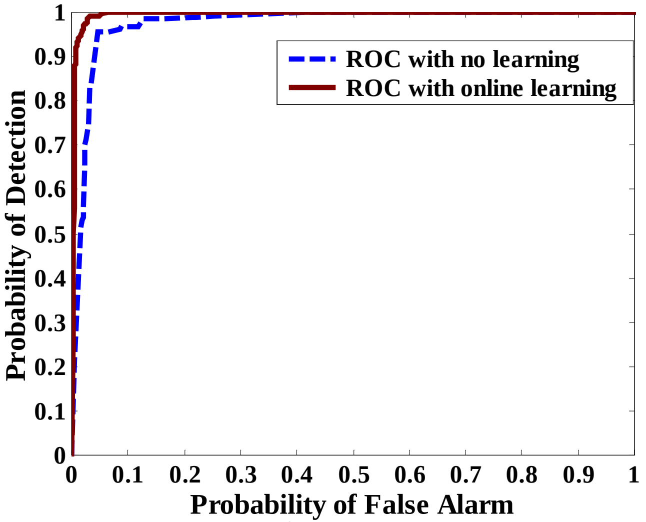


Figure 8: ROC of the email loop experiment

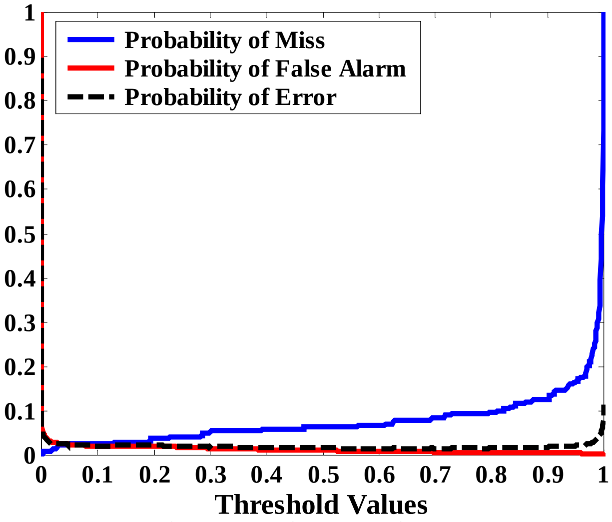


Figure 9: Probabilities of error as a function of the decision threshold

In a separate experiment, we performed batch learning over the data, testing on the same training data. The performance of the learned BN was equivalent to that of the expert-based BN. This indicates that in circumstances where batches of labeled data are available, we may be able to use that information instead of domain expertise, in order to bootstrap the system.

5. Analysis of Bayesian network contribution

One way to look at the SAC architecture is to contrast the Bayesian network layer with combinatorial ways of combining and propagating individual fault indicators. In architectures such as OpenView[29] or FireHunter[30], the fault indicators are threshold violations, and the combination operation is usually a logical OR. We believe there are advantages to the Bayesian network combination, both qualitatively and quantitatively.

5.1. Qualitative benefits

The most direct benefit of the Bayesian network combination of health indicators is that the result is a probability, rather than just a binary ok/not-ok indicator (or a tertiary green/yellow/red). Clearly, one can derive an n-ary indicator from the probability by thresholding it, and this is what we did in the anomaly detection experiment described in section 3, but the thresholding operation is not required. One direct advantage of having probabilities as the output is the ability to choose the trade-off between accuracy (false alarm rate) and sensitivity (miss rate). Therefore, should the context of the detection function imply a significant emphasis either on maximizing the detection rate, or on minimizing the false alarm rate, the BN layer is flexible enough to enable either choice, simply by adjusting the probability threshold. (This yields a different point along the ROC curve.) Such adjustment is not directly doable when using a logical combination of indicators.

We also believe that the probability fundamentally contains more information than an n-ary classification, and that this will be beneficial when composing multiple SACs, but we have not demonstrated that yet.

A more subtle advantage of the Bayesian network layer is its ability to choose how much ‘attention’ to pay to each underlying evaluator (as encoded in the CPT entries), before coming up with its final health assessment. This could be approximated by providing weights for each indicator, and using a linear combination. However, the Bayesian network allows even further refinement, by allowing different degrees of ‘trust’ to be stated for each possible answer of each evaluator. Moreover, as we saw in the previous section, those trust parameters can be arrived at, both from human expertise, and from automated statistics on collected data. Practically speaking, this means that the Bayesian network combination can take advantage of accurate evaluators, while protecting itself from inaccurate ones.

Finally, the SAC architecture allows more complex network structures than just the ‘naïve Bayes’ structure we have used so far (e.g. the ‘arrows up’ classifier in [31]). Those structures could embed deeper explanatory models of the underlying service element health, and of the relationships between the evaluators, and may yield better accuracy and sensitivity.

5.2. Quantitative gains

Pragmatically, the most important question is whether the probabilistic reasoning layer (BN) improves the performance over deterministic logical combinations of the evaluators. Naturally, the performance of the BN depends on the CPT assignments. Currently, the BN layer of SAC is a naïve BN, therefore the CPT's represent the trust in the output of the evaluators, and a wrong assignment of the CPT's results in misleading health assessments. The question becomes, suppose that the CPT's are set reasonably well, does the BN improve over logical combination of the evaluators, and by how much? We show empirically that in our experiment, the gain of the BN, with expert setting of the CPT's, over logical combinations, is significant.

We compared two commonly used logical combinations of the evaluators: the logical OR of all the evaluator assessments and a majority (MAJ) count of the evaluator opinions. (This also implied reducing the evaluator outputs to a binary answer.) The first (OR) has better or equal detection rates than any single evaluator, but has more false alarms. The second (MAJ) has lower detection rates but also less false alarms. The MAJ operator can be seen as a 'softer' version of the logical AND operator, which has the fewest false alarms, but also a very low detection rate.

In order to systematically compare the BN output to a binary output, one has to choose a general threshold picking strategy, rather than an ad-hoc one. We decided to choose the threshold corresponding to the point of equal probability of false alarm and probability of miss. This yields the point on the ROC curve closest to the (0,1) ideal mentioned in section 3.5, and is the best trade-off between the two quantities (in the absence of information regarding the relative costs of a false alarm and a miss).

Table 1 shows the detection and false alarm rates for each of the four evaluators. Table 2 shows the error rates for the OR combination, MAJ combination and the BN output using the threshold just described.

Evaluator	Detection rate (%)	False alarm rate (%)
Incoming Message rate	76	30
Max same size	78	11
To/from ratio	36	22
Max sharpness	94	7

Table 1: Error rates for individual evaluators

	Detection rate (%)	False alarm rate (%)
Logical OR	100	52
Majority count (MAJ)	66	3
BN output	95	5

Table 2: Error rates for the logical combination of evaluators and BN output

The results show that the BN improves significantly the false alarm rate in comparison to the OR combination, while reducing the detection only slightly. Compared to the MAJ combination, the BN improves detection significantly, while increasing the false alarm rate only slightly. These results were consistently repeated in additional experiments we have performed on the same data, but with different evaluators.

6. Conclusions and future work

We have presented a general architecture, which aims at autonomously assessing the health of service elements in a broad sense, by adding a layer of intelligence on top of the measurement gathering and sending paradigm. That intelligence is provided by a combination of common statistical techniques, packaged in a reusable way, and a probabilistic reasoning technology (Bayesian networks). Our implementation of the architecture is largely domain independent, with a small domain dependent customization layer.

In addition, by utilizing XML to describe the generic notion of probabilistic health and using standard Web protocols (HTTP) to transport it, we enable easy peer-to-peer conversations about component health. This is a broader paradigm than the hierarchical, manager to managed element model.

We have experimented with an instance of this architecture, customized to deal with the email services domain, and targeted at the health anomalies of virus infections and email loops. The experiment, conducted on real-life data from HP firewall mail servers logs, shows satisfactory results despite the absence of specificity and sophistication of the sensors, measures and evaluators.

Our initial attempts to apply machine learning to take advantage of feedback yielded significant improvements in the accuracy of the detector, and some encouragement toward the goal of eliminating the dependence on domain experts. In addition, should the question be raised about bringing such ‘heavy machinery’ as Bayesian inference, a detailed analysis of the contributions of the Bayesian network layer showed both qualitative and material quantitative gains compared to the standard alternatives.

Therefore we believe that our proposed architecture is a promising step toward the challenges of managing large and complex services. The approach is general and valid for arbitrary service elements. The absence of a requirement for a detailed and complete model of correct behavior is an attractive aspect of this approach. The prospect of greater sensitivity and accuracy by the combination of statistics and probabilistic reasoning is compelling. The ability to reduce a potentially broad and diverse set of noisy inputs to a single number is another advantage of the approach. The domain dependent components of the architecture and with it, the attendant required customization efforts, are small.

To realize the full potential of these concepts requires more research. As well as additional experiments involving customizations of the SAC architecture to various domains, we intend to extend this work in a number of ways. We are researching the application of more sophisticated statistical and probabilistic reasoning technology, specifically learning from unlabeled data, to make the health awareness machine more accurate. We hope to further explore the self-control part of this ‘Self-Awareness and Control’ architecture. We also intend to explore what happens when we build SACs of SACs.

Acknowledgments

Sincere thanks to Alan Karp of HP Labs for key suggestions during the inception of the SAC idea, and to Marsha Duro of HP Labs for strengthening the evaluation of that idea during the gestation of this paper. We are also grateful to the IM reviewers and shepherd, Gabriel Jakobson, for improving the signal-to-noise ratio of our IM-2001 submission.

References

[1] J. Madden, CA begins shipping 'Neugent' neural network agent *eWEEK*, Dec, 1998.

[2] G. Jakobson and M.D. Weissman, Alarm correlation *IEEE Network*, pp. 52-59, Nov, 1993.

[3] G. Jakobson and M.D. Weissman, Real-time telecommunication network management: extending event correlation with temporal constraints eds. A.S. Sethi, Y. Raynaud, and F. Faure-Vincent. pp. 290-301, 1995. Proceedings of the IEEE/IFIP 4th international symposium on integrated network management IM-95 Chapman and Hall.

[4] S. Klinger, S. Yemini, Y. Yemini, D. Oshe, and S. Stolfo, A coding approach to event correlation eds. A.S. Sethi, Y. Raynaud, and F. Faure-Vincent. pp. 266-277, 1995. Proceedings of the IEEE/IFIP 4th international symposium on integrated network management IM-95 Chapman and Hall.

[5] K.R. Sheers, HP OpenView Event Correlation Services *Hewlett-Packard Journal*, pp. 31-42, Oct, 1996.

[6] Service model for email services at Sprint S. Ramanathan, D. Caswell, M. Asawa, and R. Malpani. 1997. HP Labs ISAL. Palo Alto, CA.

[7] Using service models for management of Internet services D. Caswell and S. Ramanathan. HPL-1999-43, Mar, 1999. HP Labs ISAL. Palo Alto, CA.

[8] P. Bhoj, S. Singhal, and S. Chutani, SLA management in federated environments eds. M. Sloman, S. Mazumdar, and E. Lupu. pp. 293-308, 1999. Proceedings of the IEEE/IFIP 6th international symposium on integrated network management IM-99 IEEE Publishing.

[9] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*, San Francisco, CA: Morgan Kaufmann, 1988.

[10] C.S. Hood, Intelligent detection for fault management of communication networks 1996. Rensselaer Polytechnic Institute. UMI Dissertation Services.

[11] C.S. Hood and C. Ji, Proactive network fault detection ed. IEEE. 1997. Proceedings of INFOCOMM 97

[12] C.S. Hood and C. Ji, Intelligent agents for proactive fault detection *IEEE Internet Computing*, vol. 2, pp. 65-72, Mar, 1998-Apr 30, 1998.

[13] D. Heckerman, A. Mamdani, and M.P. Wellman, Real-world applications of Bayesian networks *Communications of the ACM*, vol. 38,Mar, 1995.

[14] C. Skaanning, F.V. Jensen, U. Kjærulff, P. Pelletier, L. Rostrup-Jensen, and L. Parker, Printing system diagnosis - a Bayesian network application pp. 259-265, 1998. Proceedings of the 9th international workshop on principles of diagnosis DX-98

[15] A tutorial on learning with Bayesian networks D. Heckerman. MSR-TR-95-06, 1995. Microsoft Research. Redmond, WA.

[16] HP Internet Business Solutions. Chai. 2000. http://chai.hp.com/

[17] IT-Analysis.com. The future is thin with HP's Chai. 99. http://www.it-analysis.com/99-09-30-2.html

[18] F.G. Cozman, EBayes - Embedded Bayesian networks, rel. 1998. http://www.cs.cmu.edu/~javabayes/EBayes/index.html/

[19] Microsoft DTAS. XML Belief Network File Format. 2000. http://research.microsoft.com/DTAS/bnformat/default.htm

[20] F. Provost, T. Fawcett, and R. Kohavi, The case against accuracy estimation for comparing induction algorithms 1998. Proceedings of the 15th international conference on machine learning ICML-98 Morgan Kaufmann. San Mateo, CA.

[21] X. Boyen and D. Koller, Exploiting the architecture of dynamic systems 1999. Proceedings of the 16th national conference on artificial intelligence AAAI-99

[22] N. Friedman, M. Goldszmidt, D. Heckerman, and S. Russell, Challenge: what is the impact of Bayesian networks on learning? ed. M.E. Pollack. 1, pp. 10-15, 1997. Proceedings of the 15th international joint conference on artificial intelligence IJCAI-97 Morgan Kaufmann. San Mateo, CA.

[23] D.J. Spiegelhalter and R.G. Cowell. Learning in probabilistic expert systems. In: *Bayesian Statistics 4*, eds. J.M. Bernardo, J.O. Berger, A.P. Dawid, and A.F.M. Smith. Oxford University Press, 1992.pp. 447-466.

[24] N. Murata, K.R. Muller, A. Ziehe, and S.I. Amari, Adaptive on-line learning in changing enviroments eds. G. Pavlou, N. Anerousis, and A. Liotta. pp. 599-605, 1996. Proceedings of the Advances in Neural Information Processing Systems NIPS-96 MIT Press. Cambridge, MA.

[25] Boosting and naive bayesian learning C. Elkan. CS97-557, 1997. Dept of CS&E. San Diego, CA.

[26] S. Ramachandran and R.J. Mooney, Theory refinement for Bayesian networks with hidden variables pp. 454-462, 1998. Proceedings of the 15th international conference on machine learning ICML-98 Morgan Kaufmann. San Mateo, CA.

[27] Online learning of Bayesian network parameters I. Cohen, A. Bronstein, and F.G. Cozman. HPL-2001-55 (R.1), Jun, 2001. HP Labs IMSL. Palo Alto, CA.

[28] Adaptive online learning of Bayesian network parameters I. Cohen, A.

Bronstein, and F.G. Cozman. HPL-2001-156, 2001. HP Labs IMSL. Palo Alto, CA.

[29] HP. OpenView. 2000. http://www.openview.hp.com/index.asp

[30] Agilent. FireHunter. 2000. http://www.firehunter.com/

[31] Bayesian mixture of classifiers A. Garg, V. Pavlovic, and T.S. Huang. 2001. unpublished manuscript.

[32] A. Bronstein, I. Cohen, J. Das, M. Duro, R. Friedrich, G. Kleyner , M. Mueller, and S. Singhal, Self-Aware Services: Using Bayesian networks for Detecting Anomalies in Internet-based Services eds. G. Pavlou, N. Anerousis, and A. Liotta. pp. 623-638, 2001. Proceedings of the IEEE/IFIP 7th international symposium on integrated network management IM-01 IEEE Publishing.

Biographies

Alexandre Bronstein works as a Senior Scientist at Hewlett-Packard Laboratories in the Internet and Mobile Systems Lab. He received a BS in mathematics, MS and PhD in computer science, from Stanford University, and an RC (reality check) from 7 years in Silicon Valley start-ups. His current research interests are in the area of health: system health professionally, human health as a hobby.

Ira Cohen is a PhD Student at the University of Illinois at Urbana-Champaign. Ira received a BS degree from Ben-Gurion University, Israel, and a MS degree from the University of Illinois at Urbana-Champaign, both in Electrical and Computer Engineering. His main research interests are in statistical learning, computer vision and affective computing.

Joydip Das is a Software Architect at Hewlett-Packard. He has over 10 years of experience in developing and architecting software for numerous successful products. He has worked in areas including broadband test and measurement equipment for ATM networks, and satellite test systems. His current interests are in the areas of large scale Internet computing platforms and federated e-services marketplaces. Joydip holds a Master of Engineering – Information Technology degree from Royal Melbourne Institute of Technology and a Bachelor of Engineering - Electrical, from Monash University, Melbourne, Australia.

Rich Friedrich has held several research and product development positions within Hewlett-Packard during the previous 18 years and is now a Principal Architect in the Internet and Mobile Systems Lab. He is currently leading a team researching next generation, large scale Internet computing utilities that support emerging services. He was the program co-chair for the Sixth IEEE International Workshop on Quality of Service held in May 1998. He has published extensively and is a co-inventor on multiple patents. He attended Northwestern University and Stanford University.

Gary (Igor) Kleyner is a Scientist and Software Engineer at Hewlett-Packard Laboratories. He has over 20 years of experience in academic and industrial research and development in the areas of theory of probability and statistics, mathematical physics and modeling, data mining and reliability analysis, Internet and e-commerce automation. He has numerous publications in those fields and holds 3 patents. His current interests are in the areas of large-scale computing platforms reliability and Internet data mining and analysis. Gary holds two BS degrees in mathematics and engineering, PhD and MS degrees from Saint-Petersburg Technical University and Northwestern University.

Sharad Singhal is Principal Scientist at Hewlett-Packard Laboratories. He has over 18 years of experience in industrial research in a number of areas including speech and video coding, signal processing, middleware technologies and systems management. He has published extensively in these areas, and holds 4 patents. His current research interests are in the areas of systems management for large-scale systems. Sharad holds PhD and MS degrees from Yale University, New Haven, and BTech from the Indian Institute of Technology Kanpur, India.