INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2013; **23**: 186–213
Published online in Wiley Online Library (wileyonlinelibrary.com) DOI: 10.1002/nem.1825

Efficient failure prediction in autonomic networks based on trend and frequency analysis of anomalous patterns

Hesham J. Abed,[1] Ala Al-Fuqaha,[1,*†] Bilal Khan[2] and Ammar Rayes[3]

[1]*Western Michigan University, Kalamazoo, MI 49008, USA*
[2]*John Jay College, City University of New York, New York, NY 10019, USA*
[3]*Cisco Smart Services, Cisco Systems, San Jose, CA 95134, USA*

SUMMARY

We describe an efficient failure prediction system based on new algorithms that model and detect anomalous behaviors using multi-scale trend analysis of multiple network parameters. Our approach enjoys many advantages over prior approaches. By operating at multiple timescales simultaneously, the new system achieves robustness against unreliable, redundant, incomplete and contradictory information. The algorithms employed operate with low time complexity, making the system scalable and feasible in real-time environments. Anomalous behaviors identified by the system can be stored efficiently with low space complexity, making it possible to operate with minimal resource requirements even when processing high-rate streams of network parameter values. The developed algorithms generate accurate failure predictions quickly, and the system can be deployed in sa distributed setting. Prediction quality can be improved by considering larger sets of network parameters, allowing the approach to scale as network complexity increases. The system is validated by experiments that demonstrate its ability to produce accurate failure predictions in an efficient and scalable manner. Copyright © 2013 John Wiley & Sons, Ltd.

Received 23 July 2012; Revised 26 March 2013; Accepted 2 April 2013

1. INTRODUCTION

As information technology (IT) infrastructure and its interconnecting telecommunication networks evolve over time, they become ever larger and more complex. This creates the need for a scalable fault management strategy wherein failure prediction techniques play a critical role. Failure prediction, in turn, requires a fundamental understanding, i.e. models, of the behaviors of complex heterogeneous systems. Models of such systems, in turn, require the collection and mining of vast amounts of real-time state information. In present-day networks, such performance and state information is often not directly available and must be synthesized to create models of ensemble behavior [1–3].

In light of these shortcomings, the concept of an *autonomic network* has emerged. Autonomic networking follows the concept of autonomic computing, an initiative started by IBM in 2001 [4]. We shall use this term here to refer to any network that is self-configuring and self-managing, on the basis of real-time performance and state information. Already, networks are increasingly service-aware. Network clients specify not only content pertaining to a service, but service characteristics as well. Contemporary examples include quality of service (QoS) requirements and service-level agreements (SLAs) which allow network resources to be used optimally towards service delivery. Network design is thus moving towards self-configuring and managing to overcome the shorter technology life cycle and the rapidly growing complexity of the technology, services and network infrastructure. Future networks are envisaged as autonomic and service-aware, orchestrating guarantees of reliability,

*Correspondence to: Ala Al-Fuqaha, Department of Computer Science, Western Michigan University, Kalamazoo, MI 49008, USA.
†E-mail: ala.al-fuqaha@wmich.edu

robustness and service support through self-management of communication resources and services. The over-arching research challenge for designers of next-generation networks is to understand the mechanisms by which the requisite network self-awareness may be instrumented [5–8].

Network fault management is one part of the solution to the over-arching challenge described above. The area of network fault management has been driven in part by immediate returns for service providers. Unfortunately, most existing fault management systems (to date) target a specific domain. In doing so, they are vulnerable to communication service providers who continue to add new capabilities and sophistication to their systems in order to meet demands of a growing population of users. The resulting fault management systems thus evolve to manage a multi-layered network with many legacy processing procedures.

Fundamentally, the value of intelligent fault management lies in its ability to help identify the root causes of failures, in order to facilitate effective corrective response. Typical fault management approaches use observations of the system to detect a failure, isolate (locate) the source of failure and diagnose the root cause by analyzing event traces [9]. Based on traces, system behavior is modeled as a finite state machine, in which failure events form a part of the event set. An integrated inter-domain network fault management strategy in a multi-layered network must thus process and leverage vast numbers of event stream histories to determine the roots of causally related chains of events.

Ideally, collateral damage due to network failures could be mitigated if they could be predicted *in advance* of their occurrence. Homeostasis could then be more easily maintained by taking corrective measures to avoid imminent failures *before* they occurred. A failure prediction system (FPS) would be considered successful if it exhibits highly accurate predictions, responsiveness levels suitable to real-time environments, capability for self-management via self-configuration and scalability (with respect to increases in network size, component diversity and the complexity of individual components). Intelligent agent-based network management systems have been developed to explore the possibility of such anticipatory behavior [10]. These approaches predict imminent fault occurrences using a model based on historical network performance. Simulation comparisons of reactive and proactive approaches to fault management (based on DEVS network data) have already demonstrated the huge potential benefits of the proactive stance [4].

Here, we set out with precisely the objective of designing an automated system that is capable of predicting the imminent occurrence of system failures, based on the recent trajectory of measurable system characteristics. Certainly, any such quest faces a plethora of questions at the very outset. In what follows, we present a general mathematical model and meta-architecture. The mathematical model and system devised are agnostic to the specific choice of failures and predictors.

The rest of this paper is organized as follows. Section 2 provides a review of related research literature and compares our approach with the existing literature. Section 3 presents our approach. Section 4 presents the problem in formal mathematical terms. Section 5 presents our solution approach in formal mathematical terms. Section 6 expresses the solution in terms of its constituent algorithms, providing a rationale behind their design, and an evaluation of their theoretical complexity and performance in practice. Section 7 provides a case study in which the proposed system is applied towards the prediction of network congestion. Section 8 represents a comparison of our results with other approaches presented in the literature. Finally, in Section 9 we conclude our study and discuss possible future extensions of this work.

2. RELATED WORK

In the following subsections, we provide a brief review of related research literature; then we present some of the main results presented in recent research work and potential impacts of contributions in this research area. Finally we provide an overview of our proposed approach.

2.1. Current approaches

Fault identification [11] has been studied extensively in the literature using many different approaches. Thottan and Ji [1] provide an excellent overview of the results of recent research efforts on the

DOI: 10.1002/nem

detection of anomalous behavior. In what follows we provide a possible taxonomy for these approaches, relating them to our own efforts which are presented in subsequent sections of this paper.

2.1.1. Rule-based approaches

Early work in the area of fault or anomaly detection was based on expert systems. In the expert systems approach, an exhaustive database containing the rules of behavior exhibited by a faulty system is used to determine fault occurrence [12–14]. In practice, this entails matching predefined rules of network anomalies to observed sequences of behavior. Rule-based expert systems are not only too slow for real-time applications, but also depend on prior knowledge about the fault conditions of the network [12,15,11,16]. The identification of faults in this approach depends on symptoms that are specific to a particular manifestation of a fault [1]. Examples of symptoms might be excessive utilization of bandwidth, number of open TCP connections or total throughput exceeded. Extensions of this approach include the more recent application of fuzzy cognitive maps (FCMs), which allow for intelligent modeling of the propagation and interaction of network faults [16].

2.1.2. Alarm correlation using state machines

In this second family of approaches, finite-state machines (FSM) are used to model alarm sequences that occur during and prior to failures. This is a natural approach to try, since faults are a disorder within a managed network that results in an external 'alarm'. In many approaches, a probabilistic finite state machine model is built for a known network fault using historical data collected from its occurrences. State machines serve both to detect the anomaly and also possibly in identifying and diagnosing the problem [17,18]. The sequences of alarms obtained from the different points in the network are modeled as states of an FSM, and alarms are assumed to contain information such as the device name as well as the symptom and time of occurrence. The probability of transitions between the states is estimated by conditioning on prior events [2,17,19,9] within historical failure data.

Within the state machine paradigm, researchers previously developed a system based on 'topographical proximity' [9], which leverages topographical information embedded in alarm data in order to evaluate the implausibility of mined sequences. In their research work, event correlation is used to extract significance and semantics from the wealth of alarm data generated by the network. The authors showed that overall system performance is significantly improved by considering network proximity constraints.

2.1.3. Pattern matching

In this class of approach, anomalies are considered deviations from normal behavior and are modeled in terms of variability in the network environment [20,21]. Online machine learning is used to build a traffic profile for a given network, using symptom-specific feature vectors such as link utilization, packet loss and number of collisions [9,22]. These traffic profiles are then categorized by time of day, day of week, and special days such as weekends and holidays. When newly acquired data fail to fit within some confidence interval of the developed profiles, an anomaly event is declared [1].

Within the pattern-matching paradigm, researchers previously developed a syntactic pattern recognition algorithm approach to extract frequent anomalous behaviors based on multi-scale trend analysis of *individual* network parameters. In Abed *et al*. [21], we presented an algorithm which operates on multiple timescale analysis to model frequently occurring anomalous behaviors. Performance measurements of the algorithm demonstrated that fault prediction is possible with high true positive and high true negative rates. In particular, in practice, failures usually do not occur because of change in a single parameter but rather because a set of interrelated parameters changes which jointly led to system failure. In this work, we extend the algorithms to extract frequent anomalous behaviors based on multi-scale trend analysis of *multiple* network parameters [23], using an algorithm where relationships between parameters were mined using association rule-mining techniques.

2.1.4. Statistical analysis

As the network evolves, each of the methods described above requires significant recalibration or 'retraining'. In contrast, by using statistical approaches, it is possible to continuously track the behavior of a network [24,25,16], making such recalibration more automated. Statistical analysis

DOI: 10.1002/nem

has been used to detect both anomalies corresponding to network failures as well as network intrusions, wherein each failure scenario differs in its possible manifestations and their characteristics. Such statistical approaches make it necessary to obtain a rich set of network data that cover a wide variety of network scenarios [26,27,38].

Within the statistical paradigm, researchers previously developed a method which combines rough sets with artificial neural networks to chart potential failure domains. In Al-Fuqaha *et al*. [4], we provided the ability to filter multidimensional observation data and remove irrelevant factors. This approach uses *k*-means clustering to classify the input data into groups, after which causal relationships between groups are discovered by association mining. A prototype system based on these ideas, the Rough Neural Network Adaptive Information Prediction system (RNNAIPS), demonstrated efficient short timeframe predictions on the Internet.

2.2. Potential impact

New contributions to the problem of fault prediction are of immediate consequence to many ongoing efforts in the development of *autonomic networks*. The main objective of autonomic computing and networking is to address complexity [28–30] by transferring more manual functions to involuntary control, thereby allowing surplus resources (human and otherwise) to become available to manage higher-level processes. A comprehensive state-of-the-art review in autonomic computing and communications is provided by Galis *et al.* [39,31]. Concrete prior and ongoing projects which consider autonomic networks, either directly or indirectly, include:

- The *Knowledge Plane* [5,31] proposed as 'a fundamentally different sort of network that can assemble itself given high-level instructions, reassemble itself as requirements change, automatically discover when something goes wrong, and automatically fix a detected problem or explain why it cannot do so'. The Knowledge Plane (as opposed to the Data Plane, which is responsible for packet forwarding) 'builds and maintains high-level models of what the network is supposed to do, in order to provide services and advice to other elements of the network'.
- *Foundation Observation Comparison Action Learn Reason* (FOCALE) is a network architecture that is designed for legacy devices, applications and next-generation cognitive network devices [12].
- The *Inference Plane* [39] was created in response to some of the shortcomings of the Knowledge Plane. The Inference Plane is a coordinated set of intelligent decision-making components that represent the capabilities of the computing elements being controlled, the constraints placed upon using different functions and the context in which they are being used.
- *Ambient Networks* (AN) [32] is a European Union (EU)-sponsored project that envisages the development of a software-driven network control infrastructure for wireless and mobile networks. Within it, an Ambient Control Space encompasses all the control functions in each domain, and can then be used as a control plane overlay to integrate and interoperate seamlessly across existing networks.
- *Autonomic Network Architecture* (ANA) [33] is a EU-sponsored project that aims at exploring new ways of organizing and using networks beyond legacy Internet technology, providing the architecture for networking autonomic devices.
- *4WARD* [5] is an EU-sponsored project that aims to make the development of networks and networked applications faster and easier, by including an in-network management function.
- *Programmable Network Management* [39] techniques allow software modules to be activated or injected into network components, which then execute customized tasks on the data passing through these components. This approach is especially attractive for realizing real-time activation and adaptation, making possible the deployment of real-time control loops.
- *Global Environment for Network Innovation* (GENI) [34] serves as a platform for research programs addressing serious problems facing today's Internet, including inadequate security, reliability, manageability and ability to evolve.

All of the above initiatives, and others, stand to benefit from advances in algorithms for fault prediction, such as those developed in this work.

DOI: 10.1002/nem

3. OUR APPROACH

In this work, the time-varying characteristics of the system are referred to as *parameters*, and will form the basis of failure prediction. The value fluctuations of a parameter in a specific timeframe are referred to as its *behavior*. Broadly speaking, there are two classes of behaviors:

- Ordinary behavior occurrence is not followed by any kind of failure. This behavior is indicative of normal operational status in a network and can occur frequently without referring to future defects in the system's functionality. In addition, ordinary behavior is centered near the mean behavior, because the majority of the behaviors of a reasonably reliable system are classified as ordinary behaviors. In contrast,
- Anomalous behavior occurrence is frequently followed by a system falling into a failure state [35]. This kind of behavior is sequentially linked to ordinary behavior, but develops abnormal values through time until reaching a failure state. Anomalous behaviors represent a deviation away from mean behavior. It should be noted here that we use the term "failure" to express (a priori specified) undesirable network state, and that this term is not used to imply the system is completely dysfunctional. Rather, it may refer to some well-specified performance degradation or decline in the quality of services.

A parameter's behavior may transition between these two classes over time. Network fault and performance management systems (see, for example, ITU-T Recommendation X.733, G.7710 and Q.822) typically report temporal sequences of ordinary behaviors followed by anomalous behaviors as a 'failure occurrence'. After a failure occurrence, the parameter may return to its ordinary behavior and continue to evolve normally.

In this work, our approach for anomalous behavior mining and fault prediction is based on extracting syntactic and descriptive information about frequent behaviors that precede a failure. Our main goal is to identify a general behavior characterization for each parameter and find its frequency. In our proposed approach, a parameter's behavior, which is represented as a time series signal, is divided into a number of segments in which each segment is characterized based on its local trend only. That is, within each segment we find the maximum fluctuation values, i.e. minimum and maximum values, which is called a *crest–trough* (C-T) pair. A sequence of C-T pairs is produced for each single segment; we then develop algorithms to find the frequency of each C-T sequence in optimal time and space complexity. The extraction is carried out at *multiple* timescales, to determine trends within and across *multiple* network parameters. Based on the collected general form of behaviors, other behaviors are considered a *match* if they have the same major trends in each corresponding segment and those trends occur in the same sequential order.

Our approach is unique and has the following advantages over other approaches presented in the literature:

- Our approach is designed to find the collective anomaly [36]. That is, a collection of related parameter instances is anomalous with respect to the entire parameters set. The individual parameter instances may not be anomalies by themselves, but their occurrence together with other parameters as a collection is anomalous.
- Our approach is designed to be generic and agnostic to the specific type of failure or performance degradation scenario (e.g. traffic anomalies, load balancing, routing, equipment malfunctioning and link utilization anomalies). In the research literature, most research studies proposed prediction systems for a limited type of failures or networks [37–39,19,20,18]. The limitation of our approach is related to the nature of the input data but not to the network/fault type.
- Failure prediction using threshold-based techniques has a drawback because it is not always true that reaching a threshold point means inevitable occurrence of a failure. On the other hand, our approach provides information about the current status of a behavior as well as its next expected direction via its trend.
- In the literature, many research studies have proposed to discover anomalies based on an abrupt change in the behavior only. This approach has limited types of applications, as it will not be effective with a wide range of applications, especially when failures occur after a smooth gradual

DOI: 10.1002/nem

change in behavior [40–43]. On the other hand, our approach finds the repetitive behavior in forms of interrelated sequences of C-Ts. This enables our approach to identify abrupt as well as gradual changes.

- For behavior matching, our approach does not use a complex and intensive comparison of signals based on the Euclidean distance between each point in a behavior and the corresponding point in another behavior. That comparison is not preferred because of its complexity, especially when similar behaviors could have shifted in the time of their occurrence; i.e. a behavior could occur before or after the occurrence of a similar behavior. In contrast, our approach is based on the much simpler trend analysis techniques to provide an efficient solution for the failure and performance degradation prediction problem.
- We implemented optimizations on the developed algorithms, which make them more suitable for real-time applications.

4. FORMAL PROBLEM STATEMENT

In this section, we introduce a mathematical formulation of the problem and a framework for evaluating candidate solutions.

Any time-varying characteristic of the system will be referred to as a *parameter*. A *parameter time series* $\overline{f}$, then, is simply a sequence of discrete, uniformly Δ-spaced real-world measurements of a specific parameter f, i.e.

$$\overline{f} \stackrel{\text{def}}{=} (f(n\Delta)|n \in \mathbb{Z}) \tag{1}$$

In our notation, the nth element of the series $\overline{f_n} = f(n\Delta)$. Here $n\Delta$ represents the *time* corresponding to *tick* number n. In what follows, we shall frequently switch between a viewpoint based on tick numbers, and a viewpoint based on continuous time, as it is convenient to the most succinct exposition.

We assume the existence of a function

$$w(n) \stackrel{\text{def}}{=} \begin{cases} 1 \text{ if the system is in } \textit{normal} \text{ state at time } n\Delta \\ 0 \text{ if the system is in } \textit{failure} \text{ state at time } n\Delta \end{cases} \tag{2}$$

that associates to each tick $n\Delta$ a Boolean value w(n) which captures whether the system is 'working' or not, with respect to some fixed set of criteria. The *failure* set of a system

$$F \stackrel{\text{def}}{=} \{n \in \mathbb{Z}|w(n) = 0\} \subseteq \mathbb{Z} \tag{3}$$

is the set of ticks at which a failure is observed.

For the remainder of this section, fix positive integers τ and ψ, satisfying $\psi|\tau$. We refer to τ as the *window size* and ψ as the *segment size*. For any tick $n \in \mathbb{Z}$, we define the *window* at n to be a length τ subseries of $\overline{f}$ which ends at tick n:

$$\overline{f}[n] \stackrel{\text{def}}{=} \left(\overline{f}_{n-(\tau-1)}, \ldots, \overline{f}_{n-2}, \overline{f}_{n-1}, \overline{f}_n\right) \tag{4}$$

Note that window $\overline{f}[n]$ in equation (4) is just a tuple of τ real numbers, i.e. $\overline{f}[n]_s$ is an element of $\mathbb{R}^\tau$. Later, in the implementation setting, τ will be referred to as the parameter analysis window (PAW).

Given a window $\overline{f}[n]$, we divide it into $\eta \stackrel{\text{def}}{=} \tau/\psi$ *segments*. In particular, segment s within the window at n is denoted as

$$\overline{f}[n]_s \stackrel{\text{def}}{=} \left(\overline{f}_{n-s\psi-(\psi-1)}, \ldots, \overline{f}_{n-s\psi-2}, \overline{f}_{n-s\psi-1}, \overline{f}_{n-s\psi}\right) \tag{5}$$

for each $s = 0, 1, \ldots, \eta - 1$. Note that segment $\overline{f}[n]_s$ is just a tuple of ψ real numbers, i.e. $\overline{f}[n]_s$ is an element of $\mathbb{R}^\psi$.

DOI: 10.1002/nem

Given a finite training set subset $F_0 \subset F$ of size $|F_0| = \Upsilon$, we define a $\Upsilon \times \eta$ *training matrix* (having Υ rows and η columns):

$$\mathcal{M}(F_0) = \begin{pmatrix} \overline{f}[n_1] \\ \vdots \\ \overline{f}[n_i] \\ \vdots \\ \overline{f}[n_\Upsilon] \end{pmatrix} \text{for } n_i \epsilon F_0 \tag{6}$$

The *i*,*s* entry of the training matrix $\mathcal{M}(F_0)$ is denoted $\overline{f}[n_i]_s$ and is the ψ-tuple of real numbers that form segment s of the window ending at n_i.

Our objective is to define a 'modeling procedure' $\mathcal{P}$, which takes the training matrix $\mathcal{M}(F_0)$ as input, and from this training matrix construct a classifier $C : \mathbb{R}^\tau \rightarrow [0, 1]$. The classifier C is a function that assigns to each *test* window $W \in \mathbb{R}^\tau$, a real value in the interval [0, 1]. If W is the sequence observed in the window prior to tick n_0, then the value of $C(W)$ is interpreted to be an estimate of the likelihood of failure at time $n_0\Delta$.

How might we evaluate a concrete modeling procedure $\mathcal{P}$? Given a *test set* (X_0,X_1) consisting of a set of non-failure ticks $X_1 \subset \mathbb{Z} \backslash F$ and failure ticks $X_0 \subset F \backslash F_0$, we may assess C's performance by the standard machine learning metrics below:

Measure	Definition
True positive rate	$\frac{\lvert\{n \epsilon X_1 \lvert [C(n)]=1\}\rvert}{\lvert X_1 \rvert}$
False positive rate	$\frac{\lvert\{n \epsilon X_0 \lvert [C(n)]=1\}\rvert}{\lvert X_0 \rvert}$
True negative rate	$\frac{\lvert\{n \epsilon X_0 \lvert [C(n)]=0\}\rvert}{\lvert X_0 \rvert}$
False negative rate	$\frac{\lvert\{n \epsilon X_1 \lvert [C(n)]=0\}\rvert}{\lvert X_1 \rvert}$

In addition, we shall be concerned with the time and space complexity of both the model extraction process and the complexity of the resulting classifier. Specifically, in the former, we will be interested to know the time and space complexity of $\mathcal{P}$ with respect to the size of $\mathcal{M}(F_0)$. In the latter, we shall be concerned with the time and space complexity of computing $C(W)$, for $W \in \mathbb{R}^\tau$, both with respect to the size of $\mathcal{M}(F_0)$ and with respect to τ (as it is increased uniformly throughout the mathematical model). Any assessment of system performance metrics must be weighed against these time/space costs, since these greatly influence the system's scalability.

5. MODEL DESCRIPTION

Before formally describing the classifier $C : \mathbb{R}^\tau \rightarrow [0, 1]$ corresponding to the training matrix $\mathcal{M}(F_0)$, we need to introduce certain new foundational concepts. These include the notion of a *trend line*, a similarity measure between trend lines (Section 5.1), and the process of extracting frequent behavior sequences (Section 5.2). These ingredients are then assembled to define our classifier in Section 5.3.

5.1. Trend lines

Given a segment $S \in \mathbb{R}^\psi$, trend line Λ (S) is a pair of tuples

$$\Lambda(S) = ((t_{\max}, v_{\max}), (t_{\min}, v_{\min})) \tag{7}$$

and is referred to as a crest–trough (CT) pair, since

$$v_{\max} \stackrel{\text{def}}{=} \max S \tag{8}$$

DOI: 10.1002/nem

$$t_{\max} \stackrel{\text{def}}{=} \max\{i = 0, \ldots, \psi - 1 \mid S_i = v_{max}\} \tag{9}$$

$$v_{min} \stackrel{\text{def}}{=} \min S \tag{10}$$

$$t_{min} \stackrel{\text{def}}{=} \min\{i = 0, \ldots, \psi - 1 | S_i = V_{\min}\} \tag{11}$$

Here $v_{\min}$ (resp. $v_{\max}$) is the crest 'C' (resp. trough 'T') value achieved by the parameter over segment S, while $t_{\min}$ (resp. $t_{\max}$) are the minimum (resp. maximum) times at which the minimum (resp. maximum) parameter value was manifested. Note that while $v_{\min} \leq v_{\max}$, it *need not be the case* that $t_{\min} \leq t_{\max}$.

For convenience, we define a trend line extraction function

$$\Lambda : \mathbb{R}^{\psi} \rightarrow (\{0, 1, \ldots, \tau - 1\} \times \mathbb{R})^2 \tag{12}$$

which takes an ψ-tuple of real numbers and extracts from it a trend line. We will define the *length of a trend line* using the standard Euclidean distance between C and T points in the 2D time-parameter plane:

$$|\Lambda(S)| \stackrel{\text{def}}{=} \sqrt{(t_{\max} - t_{\min})^2 \Delta^2 + (v_{\max} - v_{\min})^2} \tag{13}$$

A trend line is said to be *trivial* if it has length 0. Since $\psi \geq 2$ implies that either $t_{\min} \neq t_{\max}$ or $v_{\min}$ $v_{\max}$, it follows that $\Lambda(S)$ is necessarily non-trivial when $\psi \geq 2$. The unique trivial trend line for which $v_{\max} = v_{\min} = 0$ and $t_{\max} = t_{\min} = 0$ is called the *null CT pair*, denoted $\emptyset$. The null CT pair will be used as a sentinel value, since it is easily detected and is not the trend line of any segment (provided $\psi \geq 2$, as will certainly be the case throughout this work).

We seek to define a distance metric on the space of all trend lines, i.e. CT pairs (for fixed segment size ψ). Given two CT pairs

$$\Lambda^1 = \left(\left(t^1_{\max}, v^1_{\max}\right), \left(t^t_{\min}, v^1_{\min}\right)\right) \tag{14}$$

$$\Lambda^2 = \left(\left(t^2_{\max}, v^2_{\max}\right), \left(t^2_{\min}, v^2_{\min}\right)\right) \tag{15}$$

we take the distance between the two trend lines Λ^1, Λ^2 to be the sum of the L_2-norm (Euclidean distance) between the corresponding crests and troughs in each CT pair. Formally stated:

$$d\left(\Lambda^1, \Lambda^2\right) \stackrel{\text{def}}{=} \left\|\left(t^1_{\max}, v^1_{\max}\right) - \left(t^2_{\max}, v^2_{\max}\right)\right\| 2 + \left\|\left(t^1_{\min}, v^1_{\min}\right) - \left(t^2_{\min}, v^2_{\min}\right)\right\|_2 \tag{16}$$

Reflexivity, symmetry and transitivity of d is immediate from the fact that the L_2 norm is a metric on $\mathbb{R}$, and that d is merely the component-wise sum of two independent L_2 distances. It follows that:

Lemma 1: $\left(\left(\{0, 1, \ldots, \tau - 1\} \times \mathbb{R}\right)^2, d\right)$ is a metric space.

Frequently, we shall need a version of d that is rescaled to take into account the magnitude of the crest-to-trough distance. Towards this, we introduce:

$$d^*\left(\Lambda^1, \Lambda^1\right) \stackrel{\text{def}}{=} d^*\left(\Lambda^1, \Lambda^2\right) / \max\left\{\left|\Lambda^1\right|, \left|\Lambda^2\right|\right\} \tag{17}$$

defined on the set of all pairs of non-trivial trend lines. Formally, the set of non-trivial trend lines may be written as

$$\mathcal{T} \stackrel{\text{def}}{=} (\{0, 1, \ldots, \tau - 1\} \times \mathbb{R})^2 \backslash \{((t, v), (t, v)) \mid t = i = 0, \ldots, \psi - 1; v \epsilon \mathbb{R}\} \tag{18}$$

The reflexivity and symmetry properties of d are immediately inherited by d^* on $\mathcal{T}$. Unfortunately, d^* does not always satisfy the transitivity property when the variation in the lengths of trend lines is high. Given three non-trivial trend lines $\Lambda^1, \Lambda^2, \Lambda^3$ for which $|\Lambda^2| \gg |\Lambda^1|, |\Lambda^3|$, it may happen that $d^*(\Lambda^1,\Lambda^3) d^*(\Lambda^1,\Lambda^2) + d^*(\Lambda^2,\Lambda^3)$, causing the transitivity of d^* to fail.

DOI: 10.1002/nem

Because we do not find high variance in trend line lengths in real-world settings (to be described), we chose to work with $(\mathcal{T}, d^*)$ as the *space of non-trivial trend lines* (for fixed segment size ψ). On this space, we define a similarity measure that is inversely proportional to the distance d^*; i.e. the similarity of two trend lines is taken as

$$match_{ratio}\left(\Lambda^1, \Lambda^2\right) \stackrel{\text{def}}{=} 1 - d^*\left(\Lambda^1, \Lambda^2\right)$$

Although technically *match_ratio* is defined only on the space of pairs of non-trivial trend lines, we extend it to the space of all trend lines by declaring that if either Λ^1 or Λ^2 is trivial, then

$$match_{ratio}\left(\Lambda^1, \Lambda^2\right) \stackrel{\text{def}}{=} 0$$

The *match_ratio* is 1 match when the two CT pairs Λ^1, Λ^2 are identical, and takes a value less than 1 (not necessarily positive) to signify the degree of dissimilarity between the two CT pairs. A *match_ratio* of 0 is an indicator that the two pairs may have totally different trend directions. In this paper, if two trends have a matching ratio ≥ 0.5 (50%), we interpret it as the two CT pairs exhibiting similar trends. Towards this, we say that CT pairs Λ^1 and Λ^2 are *matching* if *match_ratio* $(\Lambda^1, \Lambda^2) \geq 0.5$ and we denote this as $\Lambda^1 \approx \Lambda^2$.

Figure 1 illustrates the *match_ratio* in several examples. A 100% matching ratio is found only when the couple of matched CT pairs have the same time values, the same length and same trend directions. The 0% *match_ratio* occurs when the two CT pairs have totally different trend directions. The 50% *match_ratio* occurs when the two CT pairs have a trend direction toward the same quadrant. The last case shows that the *match_ratio* can be negative when the two CT pairs have both different trend directions and different lengths.

Given two sequences of CT pairs (for a particular parameter)

$$S = \left(s^1, s^2, \ldots, s^h\right)$$

$$A = \left(\alpha^1, \alpha^2, \ldots, \alpha^{\eta'}\right)$$

we say that S is a *subsequence* of A (or 'S occurs in A') if there exists some index t (where $1 \leq t \leq \eta'$) such that for all j in the interval $1 \leq j \leq \eta$

$$s^j \neq \varnothing \Rightarrow s^j \approx \alpha^{j+t}$$

We will denote such a relationship as $S \preccurlyeq A$.

Having provided a definition of trend line similarity *match* (based on distance d^*, and in turn defined in terms of metric d), we return to developing more ingredients necessary to describe the procedure $\mathcal{P}$ that converts a training matrix $\mathcal{M}(F_0)$ to a window classifier $C : \mathbb{R}^\tau \rightarrow [0, 1]$.

Recall that the i,s entry of matrix $\mathcal{M}(F_0)$ is the ψ-sequence of real numbers $\overline{f}[n_i]_s$; i.e. segment number s of the window ending at tick n_i. In the previous section, we provided the operator Λ which maps ψ-sequences of real numbers into the set of non-trivial trend lines. Thus Λ may be applied element-wise to the matrix $\mathcal{M}(F_0)$. The resulting matrix, denoted $\Lambda\mathcal{M}$, has entries which are trend lines. The $\Lambda\mathcal{M}$ matrix will be referred to as the *trend line training matrix*, and within it the i,s entry will be denoted $\Lambda\left(\overline{f}[n_i]_s\right)$.

We now define the *support* of the i,s entry as the cumulative similarity between it and all the *other* entries in the trend line training matrix. Formally stated:

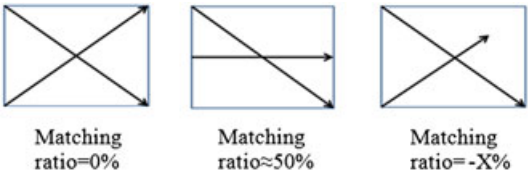

Figure 1. Illustration of the crest–trough matching algorithm

DOI: 10.1002/nem

$$\sup\left(\Lambda\left(\overline{f}[n_i]_s\right)\right) \stackrel{\text{def}}{=\!=} \sum_{\substack{i'=0 \\ i' \neq i}}^{\Upsilon-1} \sum_{\substack{s'=0 \\ s' \neq s}}^{\eta-1} match\left(\Lambda\left(\overline{f}[n_i]_s\right), \Lambda\left(\overline{f}\left[n_{i'}\right]_{s'}\right)\right) \tag{19}$$

Applying the *sup* operator element-wise to the trend line training matrix yields a $\Upsilon \times \eta$ real-valued *support matrix*, which we denote $sup(\Lambda\mathcal{M})$.

5.2. Frequent behaviors

From the support matrix $sup(\Lambda\mathcal{M})$, we define a closely related $\Upsilon \times \eta$ matrix $\beta\mathcal{M}$, which we refer to as the *frequent behaviors matrix*. In practice, the definition of $\beta\mathcal{M}$ is parametrized by a constant $sup_{\min} \geq 0$ representing the threshold of 'significant support'. Each of the $\Upsilon\eta$ entries in $\beta\mathcal{M}$ is a (possibly trivial) trend line. In particular, $\beta\mathcal{M}$ matches with $\Lambda\mathcal{M}$ precisely on those entries whose support exceeds $sup_{\min}$, and is set to the sentinel value $\emptyset$ (the null CT pair) otherwise. Formally, the row i column s entry of $\beta\mathcal{M}$ is set to

$$\beta\mathcal{M}_{i,s} \stackrel{\text{def}}{=\!=} \begin{cases} \Lambda\left(\overline{f}[n_i]_s\right) & \text{if } sup(\Lambda\left(\overline{f}[n_i]_s\right) > sup_{\min} \\ \emptyset & \text{otherwise} \end{cases} \tag{20}$$

In effect, row i of $\beta\mathcal{M}$ (denoted $\beta\mathcal{M}[i]$) is effectively the corresponding row i of $\Lambda\mathcal{M}$, thresholded so that entries of insignificant support have been replaced with the null CT pair. Each row of $\beta\mathcal{M}$ is called a *frequent behavior*. The *size* of a frequent behavior is defined to be the number of non-null CT pairs appearing within it. The size of $\beta\mathcal{M}[i]$ is denoted $\#\ \beta\mathcal{M}[i]$.

Next we introduce weights on the set of behaviors, in order to provide a measure of importance to each that is, in some sense, based on logical inference. $\beta\mathcal{M}[i]$, the frequent behavior in row i, is said to be a *refinement* of $\beta\mathcal{M}[j]$, the frequent behavior in row j, if the latter is a CT pair subsequence of the former, i.e. if $\beta\mathcal{M}[j] \preccurlyeq \beta\mathcal{M}[i]$. Note that $\preccurlyeq$ defines a *partial* order on the rows of $\beta\mathcal{M}$ —and it may frequently be the case that, given two rows, neither is a subsequence of the other.

If the frequent behavior in row i is refined by no other row, we say that row i is *elementary*. The *weight* w of elementary frequent behaviors is taken as 1. We define the weight w of a non-elementary frequent behavior j as the sum of the weights of all frequent behaviors of lesser size. To define this formally, we introduce an indicator variable:

$$m(j,k) = \begin{cases} 1 & \#\beta\mathcal{M}[k] > \#\beta\mathcal{M}[j] \text{ and } \beta\mathcal{M}[j] \preccurlyeq \beta\mathcal{M}[k] \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

and then use this to express the weight of non-elementary frequent behaviors inductively:

$$w(\beta\mathcal{M}[j]) \stackrel{\text{def}}{=\!=} \sum_{k=0}^{\Upsilon-1} m(j,k) \tag{22}$$

We define the matrix $\beta\mathcal{M}^*$ derived from $\beta\mathcal{M}$ by eliminating all rows j for which $w(\beta\mathcal{M}[j]) < sup_{\min}$. Note that $\beta\mathcal{M}^*$ is a $\Upsilon' \times \eta$ matrix of CT pairs (for some $\Upsilon' \leq \Upsilon$). The rows of $\beta\mathcal{M}^*$ represent frequent behaviors with *significant weight*.

5.3. The classifier

We now have sufficient formalism in place to describe the operation of the classifier $\mathcal{P}(\mathcal{M}\ (F_0))$. Recall that the classifier takes as input test data W, which is a tuple of τ real numbers, and must produce as output a value in the range [0, 1] which is taken as an estimate of the likelihood of imminent failure. The classifier begins by cutting W into $\eta = \tau/\psi$ disjoint segments, each consisting of ψ samples. From each of the segments W_s obtained (for $s = 0$ to $\eta - 1$), a trend line is extracted via the Λ operator. The resulting sequence of trend lines

$$\Lambda W_0, \Lambda W_1, \ldots, \Lambda Wh - 1$$

is then matched against the rows of frequent behaviors matrix $\beta\mathcal{M}$, and weighted by the behavior weights w, in order to determine the output value of the classifier.

DOI: 10.1002/nem

5.4. Selecting parameters

Until now, we have considered just one parameter. In the case of multiple parameters, an initial approach would be treated independently using the procedures described above. A more careful analysis is needed if we are to model the relationships between parameters and this is our objective in this section. Suppose we are given p distinct $\Upsilon \times \eta$ trend line training matrices (one for each of p parameters):

$$\Lambda\mathcal{M}^0, \Lambda\mathcal{M}, \ldots, \Lambda\mathcal{M}^{p-1}$$

Recall that each of the $\Lambda\mathcal{M}^h$ matrices ($h = 0, \ldots, p-1$) would be a k-row m-column matrix in which each entry is a non-trivial trend line. From this set of p trend line training matrices, we define a $\Upsilon \times p$ Boolean matrix P, which we refer to as the *parameters occurrence table* (POT). In practice, P will be parametrized by a constant $sup_{\min} \geq 0$ representing the threshold of 'significant support'. The i,h entry in the parameters occurrence table then captures whether or not *any* trend line for parameter h appeared to have significant support within the window ending at tick n_i:

$$P(i,h,sup_{\min}) \stackrel{\text{def}}{=\!=} \begin{cases} 1 & \text{if } \min_{s=1}^{\eta} sup\left(\Lambda\mathcal{M}^h[n_i]_s\right) > sup_{\min.} \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

Note that $P(i,h) = 0$ if and only if all η trend lines for parameter h in the window at n_i have support lower than $sup_{\min}$. Using P as defined above, we can define S: $\{0, \ldots, \Upsilon - 1\} \rightarrow 2^{\{0, \ldots, p-1\}}$ that serves to select parameters of significance by setting

$$S(i) \stackrel{\text{def}}{=\!=} \{h \in \{0, \ldots, p-1\} | P(i,h) = 1\} \tag{24}$$

6. ALGORITHMIC AND SYSTEM IMPLEMENTATION

In this section, we render into algorithmic terms all the elements of the formal approach described in Section 5. In the process, whenever it is appropriate, we shall highlight the principles motivating each algorithm's design, illustrate the algorithm's operation by way of example, discuss each algorithm's space and time complexity, and provide performance measurements. In Section 6.1, we provide implementation specifics of our approach to modeling historical parameter time series. In Sections 6.2 and 6.3, we apply this to the case of failure prediction based on single and multiple parameter histories. In Section 6.4, we refine the multi-parameter approach by considering correlations across multiple parameters. Finally, in Section 6.5, we put all the ingredients together to form a functional failure prediction system (FPS).

6.1. Historical parameter-based failure modeling

Throughout this exposition, Υ represents the total number of sequences (obtained from the fault management database) that are available for use as a training set. For example, Figure 2 illustrates the behavior of one parameter and $\Upsilon = 4$ different sequences that are reported to end up with a failure state. This figure shows the parameter's behavior throughout the parameter analysis window (PAW), where $\tau = 73$ ticks long. Each sequence ends in a failure state which is represented in the figure by a thick vertical line.

6.1.1. Parameter analysis window size

The above examples naturally suggest questions about the optimal selection of PAW window size. When the PAW window size is too small, significant information about the parameter's behavior is lost. On the other hand, when the PAW window size is too long, significantly more CPU and memory resources are needed to analyze the parameter's behavior. Fortunately, the optimal value for the PAW window size has been addressed by researchers, including: Thottan and Ji [1], Katzela and Schwarz [2], Ye *et al*. [44], Feather and Maxion [20] and Papavassiliou *et al*. [45]. These previous studies indicate that network anomalies occur after abrupt changes in the network's behavior, leading the network state to deviate from the normal operational states. Based on this prior work, here we exploit that abrupt changes can be used to *define* the beginning of a PAW window that is likely to contain rich information about the evolution of anomalous behaviors and how they led to failures.

DOI: 10.1002/nem

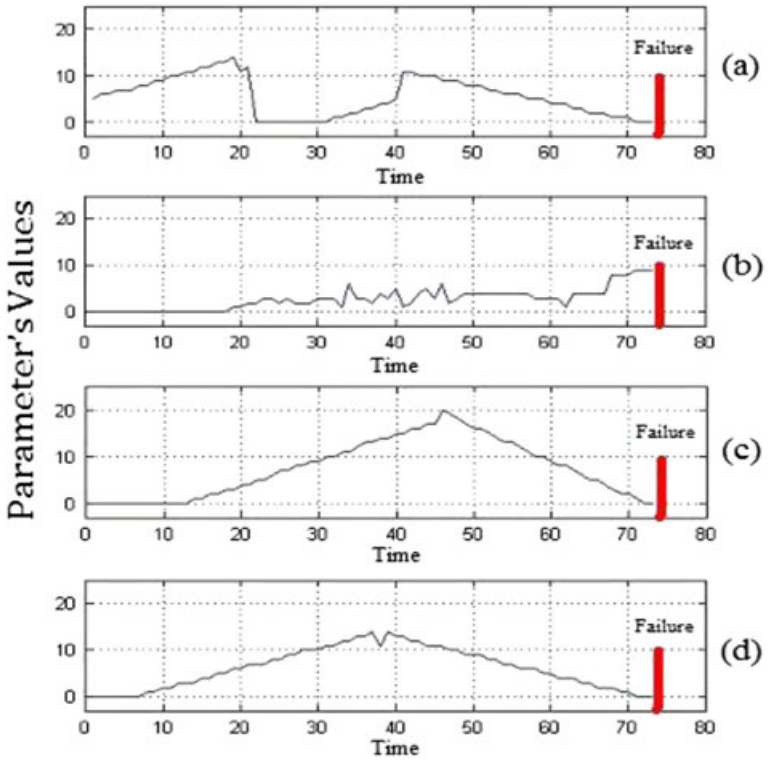

Figure 2. The behavior of one parameter in four different sequences

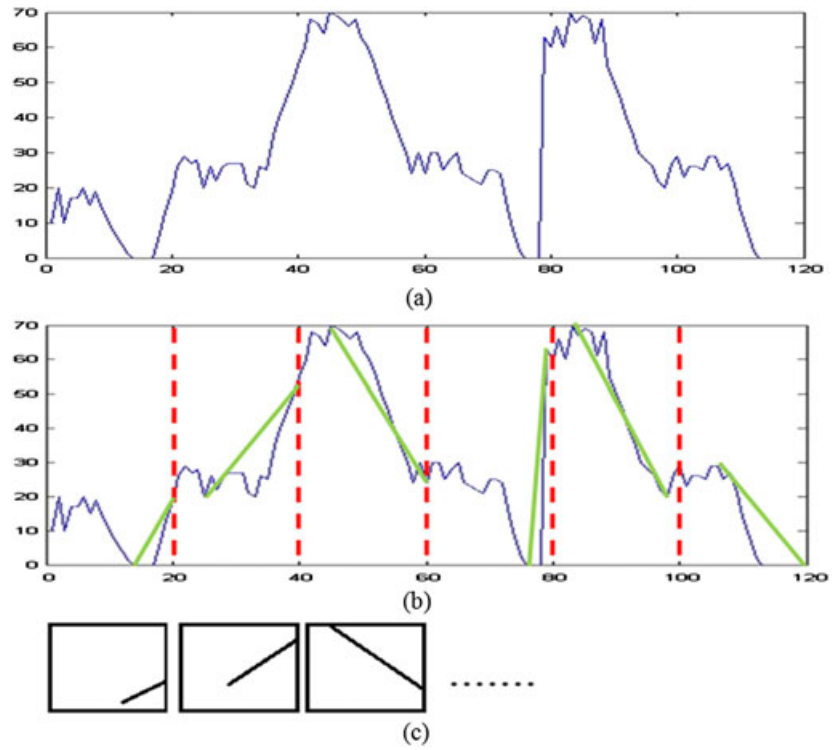

Figure 3. Representation of a parameter's behavior as a sequence of trends: (a) parameter's behavior; (b) parameter's behavior analysis using six virtual sub-windows; (c) sequence of CT pairs

DOI: 10.1002/nem

6.1.2. Trend line sequences

Following our notation, the PAW is partitioned into $\eta = \tau/\psi$ segments of size ψ. The behavior inside each segment S is represented by trend line $\Lambda(S)$, computed by determining the maximum value (crest C) and the minimum value (trough T) with the longest duration over all CT pairs. Figure 3 represents the process of trend-based analysis. In that figure, (a) represents the original parameter's behavior. In (b) we see $\eta = 6$ segments separated in the figure by dashed lines. Inside each segment a CT pair is extracted, producing a sequence of CT pairs as shown in (c), which describe the parameter behavior based on its trend analysis.

Trend lines are extracted using a preprocessing algorithm called the crest–trough pairs extraction (CTPE) algorithm, listed below.

6.1.1 CTPE Algorithm

Input: $\mathcal{M}(F_0)$ a two dimensional matrix in which each row represents a segmented window of parameter values known to have led to a failure state.
η the number segments per window.

Output: $\Lambda\mathcal{M}$ a two dimensional matrix (having Υ rows and η columns) in which each entry is a CT pairs.

Begin

Consider each row *i* column *s* in $\mathcal{M}(F_0)$, and represent the segment $\bar{f}[n_i]_s$ found therein as a crest-trough pair following the definition in Section 5.1.

End

6.2. The case of one parameter

The FABM algorithm identifies frequent behavior sequences, and is implemented as follows:

6.2.1 FABM Algorithm

Input: $\Lambda\mathcal{M}$ a two dimensional matrix (having Υ rows and η columns) in which each entry is a CT pairs.
sup_{min} minimum support threshold

Output: $\beta\mathcal{M}^*$ frequent behaviors with significant weight.

Begin

1. Find all 1-item-sequences with support ≥ sup_{min}
   - Initialize *temp*(0.. *Υ-1*) = [];
   - Initialize *count*(0.. *Υ-1*) = [0,…,0];
   - for i = 0 to *Υ-1*
     - for j = 0 to *η-1*
       - Compute $sup_{i,j}$=*sup(*$\Lambda\mathcal{M}[i,j]$) following 3.1.
       - If $sup_{i,j} \geq sup_{min}$ then
         - Append $\Lambda\mathcal{M}[i,j]$ to *temp* (i,j)
         - Increment *count(i)*
       - else // support < sup_{min}
         - Append ∅ to *temp* (i,j)
       - End if
     - End for j; End for i
2. Sort the *Υ* rows of the *temp* matrix by decreasing *count(i)*.
3. Implement the following incremental technique to calculate the support of each of the *Υ* sequences in *temp*: The first sequence in sorted *temp* is given weight *w[0]=1*. To find the weight of the j^{th} sequence in sorted *temp* we compare it against all earlier sequences *k=0..j-1*. Whenever sequence *j* is found to be a subsequence of an earlier sequence *k*, it augments its own weight *w[j]* by adding the weight of the earlier sequence *w[k]*.
4. Prune rows from *temp* whose weight w is less than sup_{min} and return the pruned matrix.

End

DOI: 10.1002/nem

6.2.1. *Guiding principles*

The FABM algorithm is designed with consideration of the following principles:

- The a priori principle is used in extracting frequent behaviors. Specifically, frequent 1-item sequences are found in step 1. This helps in finding candidate frequent sequences which may include frequent subsequences inside them.
- The sorting of sequences by descending count in step 2 minimizes the time required for finding corresponding support for each sequence. The bottom-up technique is based on the observation that p-count sequences can be detected in k-count sequences only if $k \geq p$.
- The important elements of each sequence are its frequent subsequences. Those subsequences, when they occur, exert pressure on the network status which may cumulatively lead to failures. In contrast, the infrequent subsequences can occur without noticeable effect on the network status. For this reason, the weighting process in step 3 of the algorithm discards the infrequent subsequences and focuses on frequent ones.

6.2.2. *Example*

The details of the FABM algorithm can be explained using an example as depicted in Figure 3. To simplify the example, each behavior is represented by a CT pair label. In a real-world scenario, each segment would be represented as a CT pair and these are compared by their trend values (not by their class labels) to find if they match or not. The upper left table in Figure 4 represents an input of five sequences where each sequence is represented using four CT pairs. Each row in the table denotes one sequence, while each column denotes 1-item subsequence. Suppose the $sup_{\min}$ threshold is set to 2. Step 1 of the FABM algorithm will identify C, D and A as frequent 1-item subsequences. Consequently, sequences $< C^2D^3 >$, $<A^1D^4>$, $<D^2>$, and $< A^1C^3D^4 >$ are candidate frequent sequences. In step 2, the algorithm sorts each candidate sequence based on the number of frequent items in each sequence and produces the output depicted in table (a) of the figure. In table (a), each sequence initially holds a support value of 1, which represents the current number of occurrences of the sequence. We deliberately leave blank columns to indicate the importance of the temporal information within the pattern. Next, the bottom-up technique is used to extract the support for each sequence. Starting with sequence number 2, the $< C^2D^3 >$ sequence has an occurrence in the $< A^1C^3D^4 >$ sequence based on the subsequence rule. Therefore, we add the support of $< A^1C^3D^4 >$ to $< C^2D^3 >$ and produce table (b). In table (b), starting with sequence $< A^1D^4 >$ in row 3, the algorithm finds if that sequence has occurrences in $< C^2D^3 >$ or $< A^1C^3D^4 >$. Finally, the algorithm finds the support for $< D^2 >$ in row 4 using all the sequences above it. The final result is shown in table (d), which represents all the frequent sequences that have support ≥ 2.

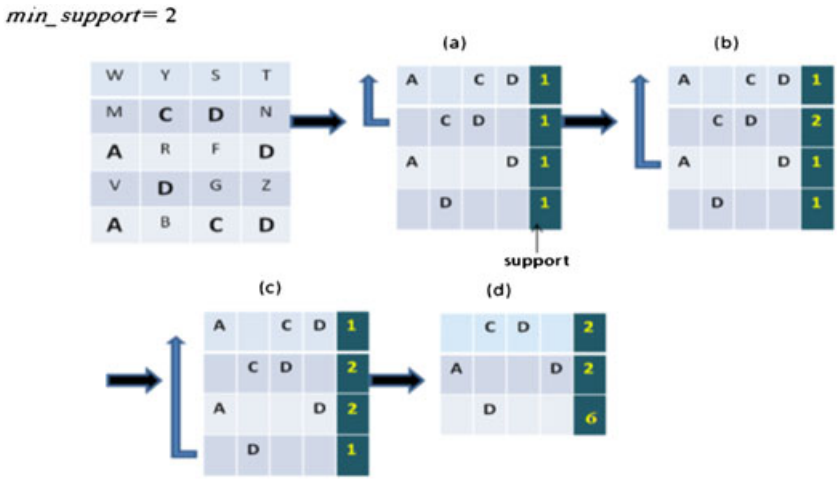

Figure 4. Illustration of the FABM algorithm

DOI: 10.1002/nem

6.2.3. *Complexity*

The time complexity of the FABM algorithm is $O(\Upsilon^2\eta^2)$. Step 1 of the algorithm requires $O(\Upsilon^2\eta^2)$ matching to extract the individual frequent segments (here $\Upsilon\eta$ is the total number of segments. In step 3, the worst case happens when half of the total numbers of the generated sequences are filled with frequent 1-item subsequences while the other half contains 1-item subsequences only. The complexity in this case is $O(\Upsilon^2\eta^2)$ because each element in the 1-item subsequences needs to be matched against the majority of segments. Each CT pair requires $O(1)$ storage. In the worst case when the segment size $\psi = 2$, the required space is $4\Upsilon\eta$ storage. Thus, FABM's space complexity is $O(\Upsilon\eta)$.

6.2.4. *Implementation issues*

One of the basic factors that affect the FABM results is the segment size. Our experimental results confirm that a smaller segment size produces low number of frequent sequences. Figure 5 shows the relationship between the segment size and the number of frequent sequences identified by FABM. In the figure, the smallest number of frequent sequences is identified when the segment size = 2. This is expected because when the segment size is very small a limited number of subsequences will be identified as frequent. While a large segment size produces more similar sequences because the small details of the behavior do not appear in the overall behavior sequence, which in turn gives an opportunity to match other behavior sequences.

Figure 6 demonstrates how a couple of behaviors, x and y, are compared in different scale levels. The x wave is depicted in the figure by a thin line, while y is shown as a thick line. The shaded area around y represents the acceptable similarity range determined by the 50% or more matching ratio.

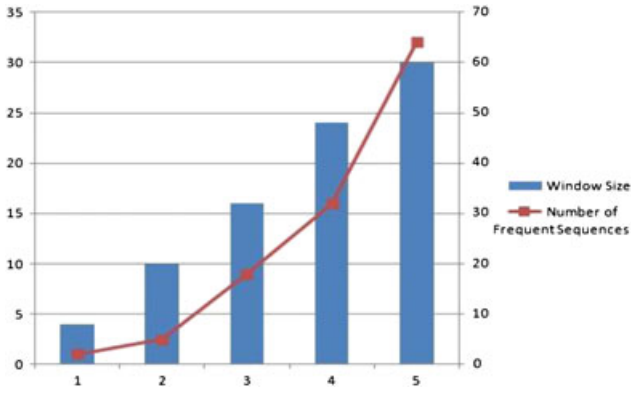

Figure 5. Relationship between window sizes and the number of corresponding frequent sequences

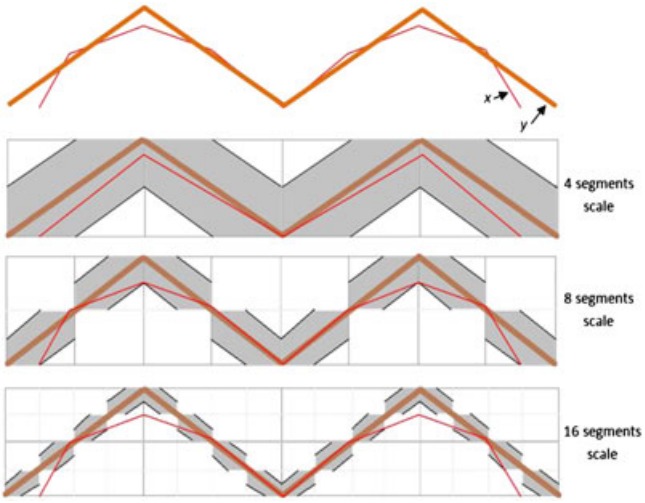

Figure 6. x and y behaviors compared in different scale levels

DOI: 10.1002/nem

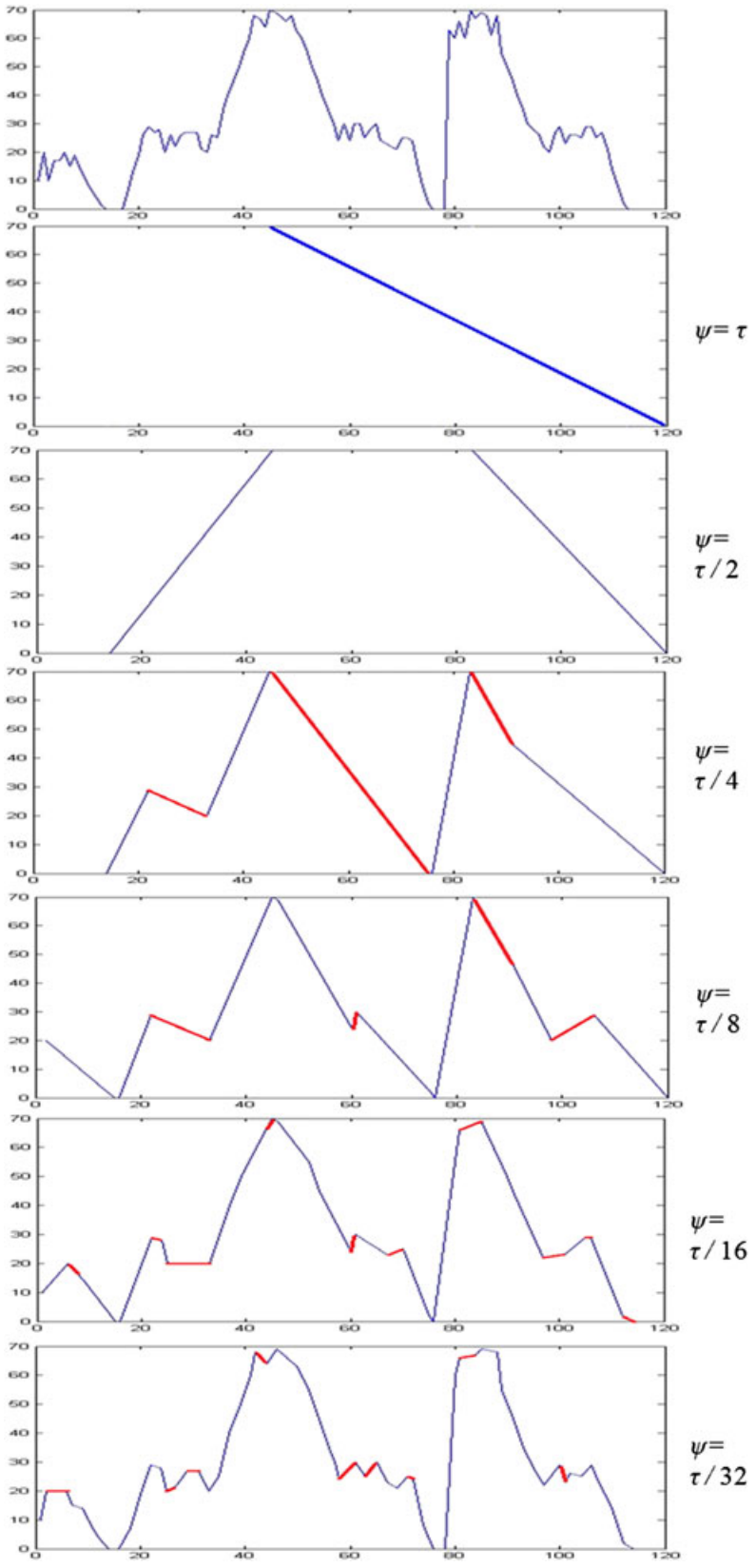

Figure 7. Multi-scale analysis of a parameter's behavior

DOI: 10.1002/nem

The figure shows that x and y are considered *similar* in the highest levels of scales (i.e. 4 and 8 segments scale in the figure), while they are *dissimilar* at 16 segments scale because some parts of x lay outside the corresponding shaded area around y. It is obvious that Figure 6 shows how a lower-level scale extract increases features and shrinks the similarity range.

Therefore, we use different segment sizes in order to extract the frequent behaviors of a parameter. This technique improves the quality of the results and does not increase the time complexity of the FABM algorithm.

A failure prediction system based on the FABM algorithm will operate at a hierarchy of timescales to identify frequent anomalous behavior. The multi-scale identification starts with a segment size ψ_0 that is of size 100% of PAW (τ). At each subsequent level, the segment size ψ_i is 1/2 of ψ_{i-1} its value at level $i-1$. At each level of the hierarchy, the FABM algorithm is used to determine frequent anomalous patterns. Figure 7 shows the parameters' behavior at different levels of the hierarchy—i.e. at different timescales. At a scale of 100% PAW, we have η=1 segment for the whole behavior and extract a single CT pair value. The scale at 50% uses $\eta = 2$ segments and so on. The smaller the scale value is, the more behavioral information is captured. Note that the red line in the figure does not represent a CT pair. It represents a virtual connection between two segments.

6.2.5. Performance evaluation

In order to assess our proposed model results, we need an inclusive database that describes a network activity in detail. Our current tests are conducted using simulation data generated by using a four-state probabilistic finite state machine (PFSM) has been built to generate the benchmark database. Each state

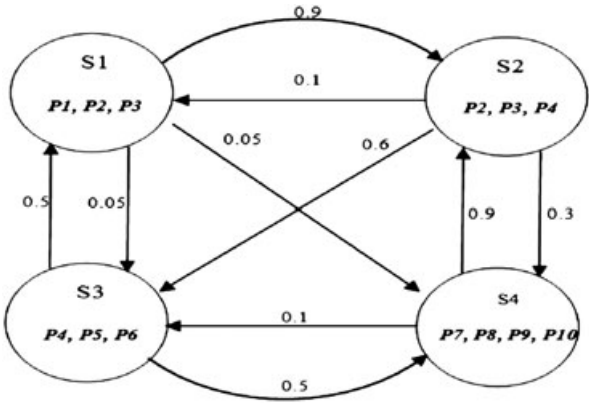

Figure 8. Probabilistic finite state machine (PFSM)

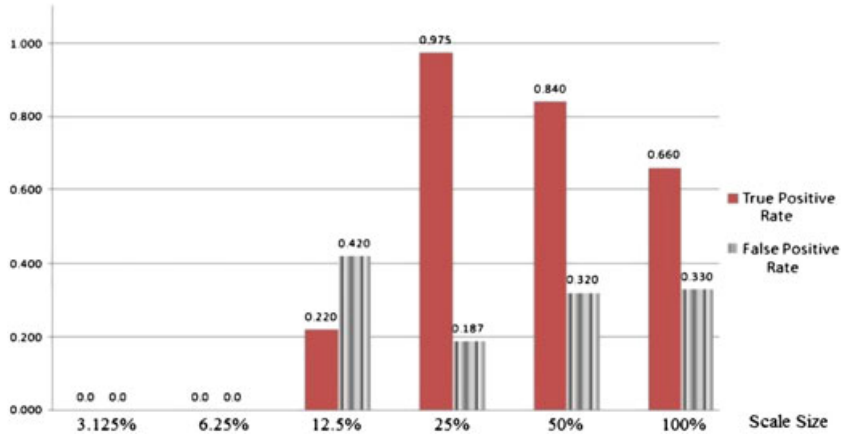

Figure 9. True and false positive rates of prediction in six timescales

DOI: 10.1002/nem

in the PFSM repeatedly produces specific type of patterns within a holding time interval. Figure 8 shows the PFSM. In Figure 8, state number 2 produces patterns that include abnormal (i.e. failure) parameter values. The other states produce patterns that include normal parameter values.

Figure 9 illustrates the true positives rate versus the false positives rate of the predication system. The figure shows that a segment size ψ of 25% $\times$ PAW yields an optimal true positives rate of 97.5% and false positives rate of 18.7%.

6.3. The case of multiple parameters

The FABMG algorithm is an enhanced or 'generalized' version of the FABM algorithm originally developed to extract frequent anomalous behaviors based on individual parameter analysis. The FABMG algorithm keeps track of each frequent parameter's behavior with respect to the PAW that has the occurrence of that behavior. Generally, this information supports the cross-parameter correlation analysis implemented using the CAAP algorithm, which we present in the next section.

6.3.1 FABMG Algorithm

Input: $\Lambda\mathcal{M}$ a 2D matrix (Υ rows, η columns) of CT pairs.
sup_{min} minimum support threshold
Output: $\beta\mathcal{M}^*$ frequent behaviors with significant weight.
Begin
1. A PAW index is assigned to each row in Υ starting with 1.
2. Find all 1-item-sequences with support ≥ sup_{min}
    Initialize *temp*(0.. *Υ-1*) = [];
    Initialize *count*(0.. *Υ-1*) = [0,…,0];
    for i = 0 to *Υ-1*
      for j = 0 to *η-1*
        Compute $sup_{i,j}$=*sup(*$\Lambda\mathcal{M}[i,j]$) following 3.1.
        If $sup_{i,j} \geq sup_{min}$ then
          Append $\Lambda\mathcal{M}[i,j]$ to *temp* (i,j); *count(i)++*
        else // support < sup_{min}
        Append ∅ to *temp* (i,j)
        End if
      End for j
    End for i
3. The result from step 2 is saved to the *temp*, where each row represents a candidate frequent anomalous sequence. Each sequence is indexed by the PAW number that contains the sequence. Sort the Υ rows of the *temp* matrix by decreasing *count(i)*.
4. Implement the following incremental technique to calculate the support of each of the Υ sequences in *temp*: The first sequence in sorted *temp* is given weight *w[0]=1*. To find the weight of the j^{th} sequence in sorted *temp* we compare it against all earlier sequences *k=0..j-1*. Whenever sequence *j* is found to be a subsequence of an earlier sequence *k*, it augments its own weight *w[j]* by adding the weight of the earlier sequence *w[k]* and performs a union operation between the $\Lambda\mathcal{M}$ indexes of both sequences.
5. Prune rows from *temp* whose weight w is less than sup_{min} and return the pruned matrix.
End

6.3.1. Example

The details of the FABMG algorithm can be explained using an example as depicted in Figure 10. In a real-world example, each subsequence should be represented with its trend values as a *crest–trough* pair, where *crest* represents the highest top and *trough* represents the lowest bottom of a behavior that lay inside an analysis window. To simplify this example, we represent each distinct CT pair by a symbol.

DOI: 10.1002/nem

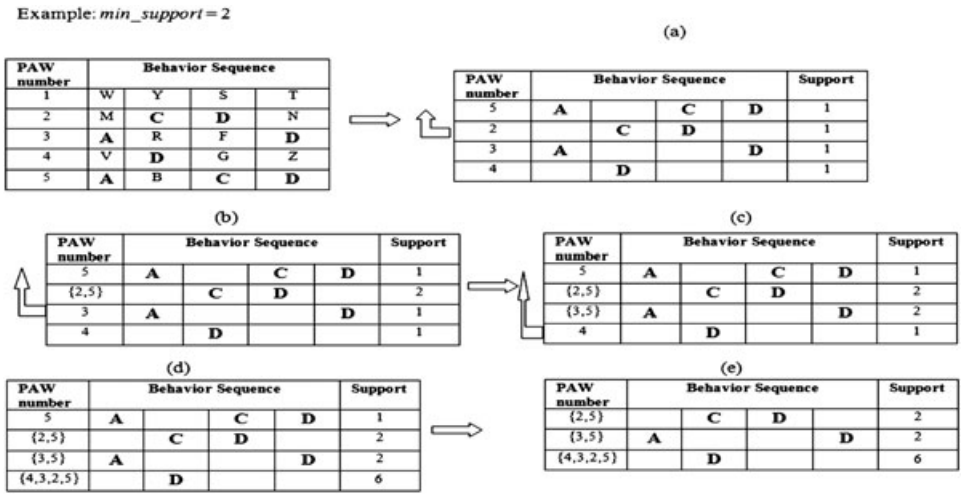

Figure 10. Illustration of the FABMG algorithm

Subsequences are compared by their symbols to determine whether they match or not. The upper left-hand table in Figure 10 represents an input of five sequences, where each sequence is represented using four CT pairs. Each row in the table denotes one sequence, while columns 2–5 denote 1-item sets. The first column stands for the PAW number of each sequence. Suppose the *min_support* threshold is set to 2. Now step 2 of the FABMG algorithm will identify C, D, and A as frequent 1-item sets. Consequently, sequences $< C^2D^3>$, $<A^1D^4>$, $<D^2>$, and $< A^1C^3D^4 >$ are candidate frequent sequences. In step 3, the algorithm sorts each candidate sequence based on the number of frequent items in each sequence and produces the output depicted in table (a) of the figure. In table (a), each sequence initially holds a support value of 1, which represents the current number of occurrences of the sequence. We deliberately leave blanked columns to indicate the importance of the temporal information within the pattern. Next, the bottom-up technique is used to extract the support for each sequence. Starting with sequence number 2, the $< C^2D^3 >$ sequence has an occurrence in the $< A^1C^3D^4 >$ sequence based on the subsequence rule. Therefore, we add the support of $< A^1C^3D^4 >$ to $< C^2D^3 >$ and append the PAW's number of the first sequence to the PAW's numbers of $< C^2D^3>$, results shown in table (b). In table (b), the algorithm finds if that sequence has occurrence in $< C^2D^3 >$ and $< A^1C^3D^4 >$ starting with sequence $A^1D^4 >$ in row 3. Finally, the algorithm finds the support for $< D^2 >$ in row 4 using all the sequences above it. The final result is shown in table (d), which represents frequent sequences that have support ≥ 2.

6.3.2. Complexity

The FABMG algorithm has the same polynomial computational complexity of $O(\Upsilon^2\eta^2)$ as in FABM. One can see this by first noting that step 2 of the algorithm requires $O(\Upsilon^2\eta^2)$ matching process to extract the individual 1-item frequent subsequences, where n is the total number of subsequences. Furthermore, in step 4, the worst case occurs in two scenarios: first, when half of the total numbers of the generated sequences are filled with frequent 1-item sets while the other half contains 1-item sets only; second, all of the generated sequences are similar and filled with frequent 1-item sets. The complexity in those cases is $O(\Upsilon^2\eta^2)$ since each element in the 1-item set needs to be matched against the majority of the behaviors. It follows that the time complexity of FABMG is of $O(\Upsilon^2\eta^2)$.

6.4. Accounting for correlations across parameters

In this section we show how an algorithm based on association rules mining can be used to relate multiple network parameters, thereby revealing the relationships between frequent anomalous behaviors. The proposed correlation analysis across parameters algorithm (CAAP) operates as follows:

DOI: 10.1002/nem

6.4.1 Correlation Analysis Across Parameters Algorithm
Input: $\Lambda\mathcal{M}$ a two dimensional matrix (having Υ rows and η columns) in which each entry is a CT pairs.
sup_{min} minimum support threshold
Output: sets of correlated parameters S
Begin
1- for i = 1 to number of parameters
for each parameter, get behavior table T_i; via FABMG.
end for
2- Create the Parameters Occurrence Table (POT) which is a two dimensional table where the rows represent the PAW numbers and the columns represent the parameters. Each element in the table has either 0 to indicate no occurrence or 1 to indicate an occurrence of a particular parameter. The table is created as follows:
2.1 for each T_i, perform union operation between all the sets that are included in the PAW number column and let G hold the result.
2.2 for each PAW number in G, mark the corresponding (PAW, parameter) in POT with 1 to indicate the occurrence of parameter number i in a particular PAW.
3- for r = 1 to the number of rows in POT
for j = 1 to the number of parameters in POT
if POT (r, j) == 1 then let S(r) = S(r) U P_j ;
end if ; end for j ; end for r
4- Call Association rules mining algorithm
End

6.4.1. Guiding principles

The CAAP algorithm starts by calling the FABMG algorithm for each parameter in order to get the frequent anomalous sequences for each individual parameter. In step 2, CAAP creates the parameters occurrence table (POT), which is a two-dimensional table in which the rows represent the PAW numbers, while the columns represent the parameters.

The use of multiple parameters analysis improves the quality of prediction. This improvement can be understood by considering a setting where $E_1, E_2, \ldots, E_n$ are a set of n events included in parameters and $F_1, F_2, \ldots, F_m$ are a set of m failures. The causal relationship between the events and failures can be represented as

$$P\left(F_f|E_a, E_b, \ldots, E_x\right) > 0 \tag{25}$$

and individually as

$$P\left(F_f|E_a\right) > 0, P\left(F_f|E_b\right) > 0, \ldots, P\left(F_f|E_c\right) > 0$$

where $1 \leq a,b,x \leq n$ and $1 \leq f \leq m$. That is, $E_a, E_b, \ldots, E_x$ are jointly related and lead to F_f failure, while each event considered individually contributes to the likelihood of failure occurrence. Note that equation (25) also represents a single-level Bayesian network, where $E_a, E_b, \ldots, E_x$ are the parent nodes and F_f is the child node. It follows that in equation (25), considering a larger number of interrelated events leads to more accurate prediction of failure F_f.

6.4.2. Association rules [16]

The association rule mining is defined by Agrawal *et al.* as: Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n binary attributes called *items*. Let $D = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the *database*. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A *rule* is defined as an implication of the form $X \Rightarrow YX \Rightarrow Y$, where $X, Y \subseteq IX, Y \subseteq I$ and $X \cap Y = \varnothing X \cap Y = \varnothing$ [3]. To extract interesting rules, constraints on various measures of significance and interest can be used. The best-known constraints are minimum thresholds on support and confidence [16,3]. The support

DOI: 10.1002/nem

(a)

	P_1	P_2	P_3	P_4	P_5
PAW_1	0	0	0	1	1
PAW_2	1	1	1	1	0
PAW_3	1	1	1	0	1
PAW_4	1	1	0	0	1
PAW_5	1	1	1	1	1

(b)

S(1)	P_4P_5
S(2)	$P_1P_2P_3P_4$
S(3)	$P_1P_2P_3P_5$
S(4)	$P_1P_2P_5$
S(5)	$P_1P_2P_3P_4P_5$

By conducting association analysis on S, we extract the correlation relationships between parameters:

(c)

Association rules	support	Association rules	support	Association rules	support	Association rules	support
P_1P_2	80%	P_2P_5	60%	$P_1P_2P_5$	60%	$P_3P_4P_5$	20%
P_1P_3	60%	P_3P_4	40%	$P_1P_3P_4$	40%	$P_1P_2P_3P_4$	40%
P_1P_4	40%	P_3P_5	40%	$P_1P_3P_5$	40%	$P_1P_2P_3P_5$	40%
P_1P_5	60%	P_4P_5	40%	$P_1P_4P_5$	20%	$P_1P_2P_4P_5$	20%
P_2P_3	60%	$P_1P_2P_3$	60%	$P_2P_3P_4$	40%	$P_1P_3P_4P_5$	20%
P_2P_4	40%	$P_1P_2P_4$	40%	$P_2P_3P_5$	40%	$P_2P_3P_4P_5$	20%
				$P_2P_4P_5$	20%	$P_1P_2P_3P_4P_5$	20%

Figure 11. Illustration of the CAAP algorithm

$sup(X)$ of an item set X is the proportion of transactions in the dataset which contain the item set, while *confidence* of the rule $conf(X \Longrightarrow Y)$ is defined as support of $X \cup Y$ over the support of X.

6.4.3. *Example*

The details of the CAAP algorithm can be explained using an example as depicted in Figure 11.

Table (a) in Figure 11 illustrates an adjacency matrix that represents the relationships between each parameter and the PAW that has the occurrence of that parameter. That is, the first column in table (a) represents P1 occurrence. It expresses that P1 has an occurrence in PAW2, PAW3, PAW4 and PAW5. In contrast, P1 does not have frequent occurrence in PAW1. Each ZERO value in the table indicates no relationship, while ONE value indicates existence of a relationship. Note that table (a) of Figure 11 can be generated based on the FABMG analysis results for each parameter. That is, the first column in table (a) of Figure 11 is an interpretation of table (e) in Figure 10 when the results in Figure 10 represent the analysis of parameter 1 behavior. Table (b) of the same figure is extracted from table (a). For each ONE value in the same row, a set of interrelated parameters is created. That is, the first set S1 includes P4 and P5. The number of sets in table (b) is equal to the number of PAWs in table (a). After creating the sets in table (b), a straightforward process that leads to discovery of the association rules and their support is implemented as shown in table (c). Table (c) can be generated by using association rules mining algorithms that are used to discover the rules from a given group of sets. Note that in (c) we only list the left-hand side (LHS) of the extracted association rules. The right-hand side (RHS) refers to a particular *failure* (omitted from the figure).

6.4.4. *Performance evaluation*

Our experimental results depicted in Figure 12 show the relationship between the number of correlated parameters and the accuracy of prediction expressed by the total number of true positive and true

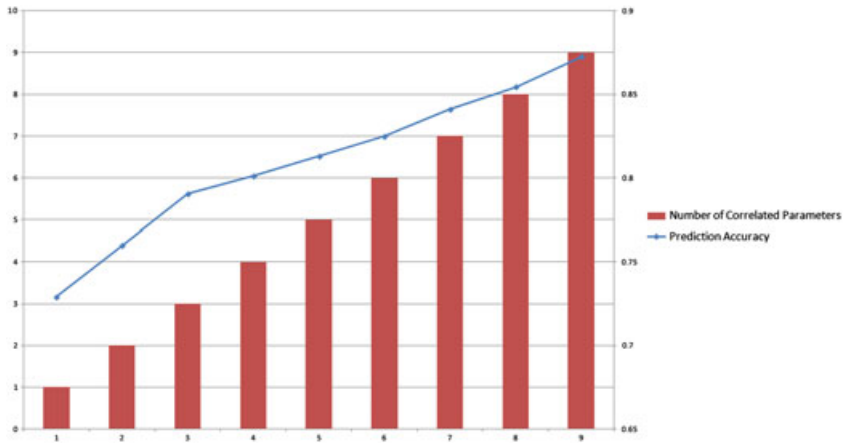

Figure 12. The relationship between correlated parameters and prediction accuracy

DOI: 10.1002/nem

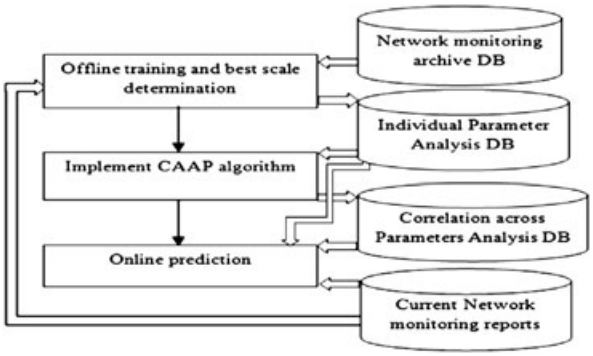

Figure 13. Illustration of the prediction system FPS

negative predictions. Figure 12 demonstrates that considering the interrelated parameters improves the quality of prediction.

6.5. *The failure prediction system*

This section presents a failure prediction system based on using the FABMG and CAAP algorithms; its three major stages are shown in Figure 13:

- *Offline training and best scale determination*. In this stage, the FPS system processes the *network monitoring archive database* (DB) to train its classifier. This stage starts by identifying failure events inside the DB then implements the individual parameter analysis using the FABMG algorithm. The results are stored in the *individual parameter analysis DB*, which is similar to the information depicted in Figure 10 table (e). In order to determine the best timescale, i.e. segment sizes that produce the best possible prediction results, different timescales are considered and the prediction quality of each is measured. Figure 9 illustrates the true positives rate versus the false positives rate of the predication system. Based on the figure, we see that a segment size that is

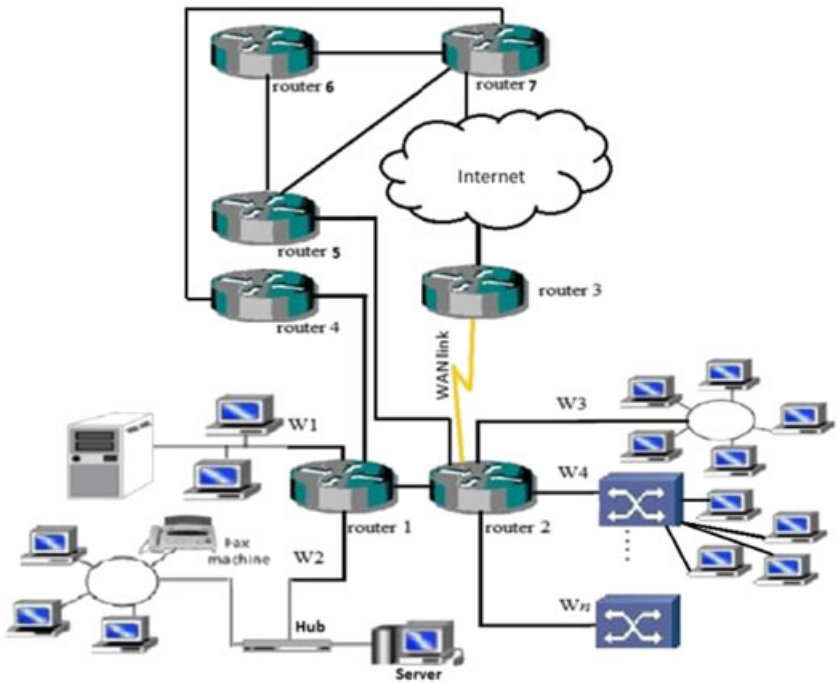

Figure 14. WAN network

DOI: 10.1002/nem

25% of the PAW provides the best results, achieving a true positive rate of 97.5% and a false positive rate of 18.7%. Note that the prediction results in this stage also help us to assess whether the training data in the archive DB span sufficient breadth of failure scenarios. Because of this fact, this stage is typically executed under human supervision since it may require extending the training set.

- *Implement CAAP algorithm.* In this stage, the correlation across parameters analysis is conducted. The result is called the *correlation across parameters analysis DB*, which consists of a set of association rules and represents the frequent occurrence of a number of parameters jointly, in a data structure similar to that shown in Figure 11(c).
- *Online prediction*. This stage takes the current network monitoring reports and predicts the network's imminent status, based on the training information in the correlation across parameters analysis database.

7. A CASE STUDY: PREDICTING NETWORK CONGESTION

The following case study concerning network congestion is used to clarify the benefits of the proposed approach. *Network congestion* occurs when incremental increases in network offered load lead either to a small increase in network throughput or to an actual reduction in network throughput. Typical ramifications include blocking of new connections, queuing delays or packet losses. Figure 14 represents n LANs interconnected by two routers. The LANs on the left-hand side utilize router1 and the LANs on the right-hand side utilize router2.

A failure occurs when the throughput rate in router 1 drops down because of congestion. For example, a failure or performance degradation condition can be generated when router 1 has a queue length that is more than 80% filled. Such a condition can lead to overwhelming the router with traffic and result in deteriorated QoS. The individual parameter analysis in this case study shows causal relationships between single parameters and the failure as follows:

- High offered load on link W1 (HOL (W1)) → failure
- HOL (W2) → failure
- HOL (router 2) → failure
- HOL (router 4) → failure
- Long queue buffer length in router1 (LQBL (router 1)) → failure

We see from this test case that using the individual parameter analysis is not sufficient to predict the failure accurately. The strongest relationship exists between the failure and the LQBL event because, by definition, LQBL is a symptom of the failure—but LQBL occurrence does not always lead to the failure. As we shall see, considering more interrelated parameters lead to improves the quality of prediction. Using the multiple parameter analysis presented in this paper, we get the following results:

- HOL (W1) AND LQBL (router 1) → failure
- HOL (W2) AND LQBL (router 1) → failure

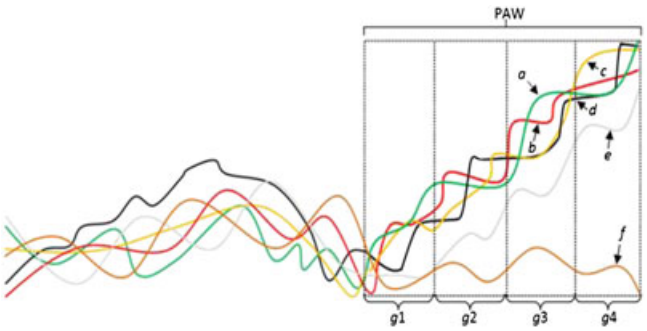

Figure 15. Offered load on router 2

DOI: 10.1002/nem

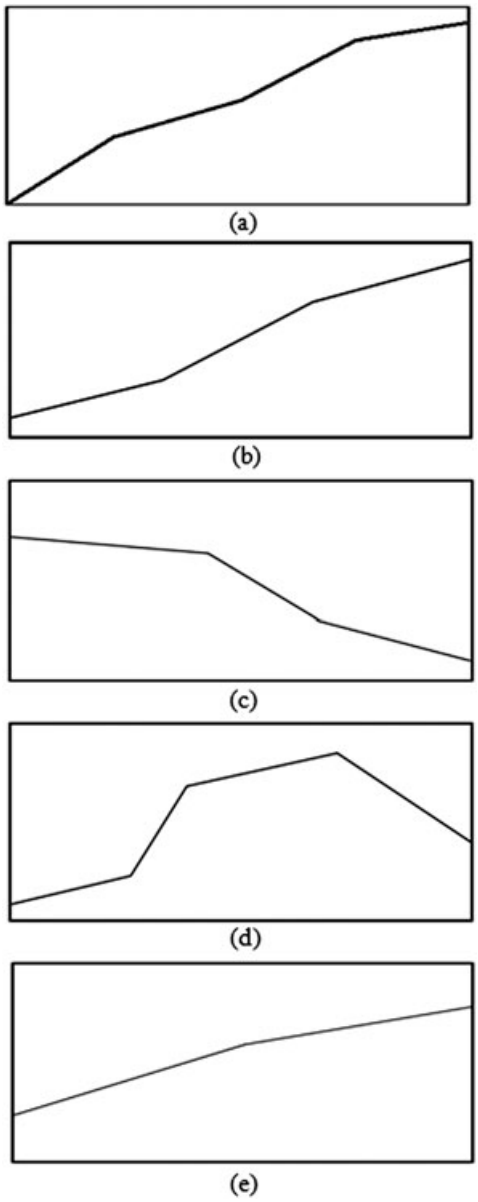

Figure 16. (a) Frequent C-T sequence for offered load on router 2. (b) Frequent C-T sequence for router 1's queue buffer length. (c) Frequent C-T sequence for the decline in router 1 throughput rate. (d) Frequent C-T sequence offered load on router 4. (e) Frequent C-T sequence offered load on link W1

- HOL (W1) AND HOL (W2) → failure
- HOL (W1) AND HOL (router 2 OR router 4) → failure
- HOL (W2) AND HOL (router 2 OR router 4) → failure
- HOL (router 2 OR router 4) AND LQBL (router 1) → failure
- HOL (W1) AND HOL (W2) AND LQBL (router 1) → failure
- HOL (W1) AND HOL (router 2) AND LQBL (router 1) → failure
- HOL (W2) AND HOL (router 2) AND LQBL (router 1) → failure
- HOL (W1) AND HOL (W2) AND HOL (router 2 OR router 4)
- AND LQBL (router 1) → failure

In the last results, the failure can be predicted with strong support when two or more of the antecedent events occur jointly. The strongest support can be found when all the antecedent events occur at the

DOI: 10.1002/nem

Table 1. Comparison of our proposed technique with other prediction techniques proposed in the literature

Prediction technique	Precision	Recall	F-measure
HSMM	0.852	0.657	0.7419
Similar event prediction	0.8	0.923	0.8571
FABM	0.839	0.975	0.902
FABMG	0.875	0.98	0.9245

same time. Consequently, our approach for online prediction starts by looking at the largest number of antecedent parameters and checks their occurrence in the current network activities. If such correlated parameters are found, the FPS system generates an alarm indicating that a failure condition is predicated.

Using a simulation scenario could make the previous theoretical analysis more realistic. Figure 15 shows samples of the behavior for the offered load on router 2 link prior to router 1 failure. It has been determined by the failure prediction systems that best scale level is when the number of segments is 4; g1, g2, g3 and g4 refer to segment number 1, 2, 3, and 4 correspondingly. Figure 15 shows that behavior *f* is infrequent because the majority of behaviors follow different trends. Based on the information in Figure 15, the most frequent CT sequence is depicted in Figure 16(a). Similarly, Figure 16(b) shows the frequent CT sequence for router 1's queue buffer length. Figure 16(c) represents how the decline in router 1 throughput rate occurs simultaneous to other parameter behaviors. Figure 16(d) depicts the offered load on router 4 link, and Figure 16(e) represents offered load on link W1. Offered load on link W2 is not frequent and therefore is not considered here.

8. RESULTS COMPARISON

Comparing our results with alternative techniques is not an easy task in the absence of a general framework for assessment and validation. Therefore we compared our results with two other approaches [47,48] to demonstrate the strengths of our proposed approach. Table 1 summarizes the precision, recall and F-measure metrics of our proposed FABMG approach and how it compares to the similar event prediction [47] and HSMM [48] approaches. It is evident that our proposed approach is scoring higher on all three measures.

Precision and recall are utilized to compute the steady-state system availability [47]. Precision can be found using equation [47]:

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \tag{26}$$

Recall is also called the true positive rate and is calculated as follows:

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}} \tag{27}$$

For clarification, consider the following example. If a prediction algorithm achieves precision of 0.9, the probability is 90% that any generated failure warning is correct (refers to a true failure) and 10% are false positives. A recall of 0.8 expresses that 80% of all true failures are predicted (and 20% are missed).

A widely used metric that integrates the trade-off between precision and recall is the F-measure [47]:

$$\text{F} - \text{measure} = \frac{2 \ \text{Precision} \ \text{Recall}}{\text{Precision} + \text{Recall}} \in [0, 1] \tag{28}$$

It should also be emphasized that our approach requires $O(n^2)$ run time complexity while the other two techniques presented elsewhere[47,48] are more complex, especially the HSMM, which is developed based on hidden semi-Markov models.

DOI: 10.1002/nem

9. CONCLUSIONS AND FUTURE WORK

In this work, we presented new data-mining techniques that extract essential models of anomalous behavior sequences known to be precursors of system failure conditions. The technique leverages the discovered correlations across multiple system parameters to improve its prediction accuracy. The proposed prediction system (FPS) uses the FABMG and CAAP algorithms, and culminates in an effective online failure prediction based on a classifier generated with reference to a training corpus. The FPS presented provides accurate predication results, while maintaining the polynomial time complexity of its algorithms and linear space complexity.

In this paper, we focus on one part of network management: namely, prediction of failures and performance degradations. We mainly propose that one can design a system to achieve homeostasis through using the FPS system presented in the paper. Remarkably, the FPS is designed to accept input as general as possible to target various kinds of failure and performance degradation scenarios. The core of the prediction process is based on using the most frequent parameter behaviors that take place prior to a failure occurrence.

In the future, we intend to extend the FPS to allow it to update the anomalous sequence weights in its individual parameter analysis DB based on its prediction performance. Such an online learning strategy could improve the quality of predictions, enable it to adapt to shifts in network function, and enable it to be initialized using smaller training sets. We also intend to address the problem of determining optimal thresholds for minimum support $sup_{\min}$ and the *match_ratio* cutoff. We also intend to explore the extent to which solutions may leverage general-purpose graphics processing units (GPGPU) to implement the proposed system in a parallel environment, providing improved response times and scalability.

The approach presented in this work can be integrated with any network using an independent computer/entity that monitors the functions of other networks entities and utilizes our proposed algorithm to predict failure and performance degradation conditions. The decision on the course of actions that need to be taken when a failure or performance degradation condition is detected is left to higher-level applications.

REFERENCES

1. Thottan M, Ji C. Anomaly detection in IP networks. *IEEE Transactions on Signal Processing* 2003; **51**(8): 2191–2204.
2. Katzela I, Schwarz M. Schemes for fault identification in communication networks. *IEEE/ACM Transactions on Networking* 1995; **3**: 753–764.
3. Medhi D, Jain S, Shenoy Ramam D, Thirumalasetty SR, Saddi M, Summa F. A network management framework for multi-layered network survivability: an overview. In *IEEE/IFIP Conference on Integrated Network Management*, May 2001; 293–296.
4. IBM. Autonomic Computing: IBM's Perspective on the State of Information Technology (2001). Available: http://www.ginkgo-networks.com/IMG/pdf/autonomic_computing.pdf [last accessed 12 May 2013].
5. Galis A, Denazis S, Bassi A, Giacomin P, Berl A, Fischer A, DeMeer H, Srassner J, Davy S, Macedo D, Pujolle G, Loyola JR, Serrat J, Lefevre L, Cheniour A. Management architecture and systems for future Internet networks. In Towards the Future Internet. IOS Press, 2009; 112–122.
6. Lakhina A, Crovella M, Diot C. Diagnosing network-wide traffic anomalies. In *Proceedings of ACM SIGCOMM*, August 2004.
7. Lazar A, Wang W, Deng R. Models and algorithms for network fault detection and identification: a review. In *Proceedings of IEEE International Conference on Communications*, 1992.
8. Li X, Bian F, Crovella M, Diot C, Govindan R, Iannaccone G, Lakhina A. Detection and identification of network anomalies using sketch subspaces. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, 2006.
9. Al-Fuqaha A, Rayes A, Kountanis D, Abed HJ, Kamel A, Salih R. Prediction of performance degradation in telecommunication networks using joint clustering and association analysis techniques. In *IEEE Globecom*, 2010.
10. Nair MK, Gopalakrishna V. Agent based web services with RuleML for network management. In *NETCOM '09: First International Conference on Networks and Communications*, 2009.
11. Saple A, Yilmaz L. Agent-based simulation study of behavioral anticipation:anticipatory fault management in computer networks. *ACM SE'06*, 2006, Melbourne, FL.
12. Ndousse TD, Okuda T. Computational intelligence for distributed fault management in networks using fuzzy cognitive maps. In *Proceedings of IEEE ICC*, Dallas, TX, June 1996; 1558–1562.
13. Lewis L. A case based reasoning approach to the management of faults in communication networks. In *Proceedings of IEEE INFOCOM*, Vol. 3, San Francisco, CA, March 1993; 1422–1429.

DOI: 10.1002/nem

14. Mountzia MA, Rodosek GD. Using the concept of intelligent agents in fault management of distributed services. *Journal of Network and Systems Management* 1999; **7**(4): 425–446.
15. Franceschi AS, Kormann LF, Westphall CB. Performance evaluation for proactive network management. In *Proceedings of IEEE ICC*, Dallas, TX, June 1996; 22–26.
16. Casas P, Mazel J, Owezarski P, UNADA: unsupervised network anomaly detection using sub-space outliers ranking. In *Proceedings of the 10th international IFIP TC 6 Conference on Networking*, Part I, May 2011.
17. Rouvellou I, Hart G. Automatic alarm correlation for fault identification. In *Proceedings of IEEE INFOCOM*, Boston, MA, April 1995; 553–561.
18. Frohlich P, Nejdl W. Model-based alarm correlation in cellular phone networks. In *Proceedings of the International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS)*, January 1997.
19. Bouloutas A, Hart G, Schwartz M. On the design of observers for failure detection of discrete event systems. In Network Management and Control. New York: Plenum, 1990.
20. Feather F, Maxion R. Fault detection in an Ethernet network using anomaly signature matching. In *Proceedings of ACM SIGCOMM*, Vol. 23, San Francisco, CA, September 1993; 279–288.
21. Abed HJ, Al-Fuqaha A, Guizani M, Rayes A. Failure prediction based on multi-scale frequent anomalous behavior identification in support of autonomic networks. In *IEEE Globecom*, 2010.
22. Lakhina A, Crovella M, Diot C. Mining anomalies using traffic feature distributions. In *Proceedings of ACM SIGCOMM*, August 2005.
23. Abed HJ, Al-Fuqaha A, Aljaafreh A. Failure prediction based on multi-parameter analysis in support of autonomic networks. In *IEEE ICCIT*, 2011.
24. Gu X, Wang H. Online anomaly prediction for robust cluster systems. In *IEEE 25th International Conference*, 2009.
25. Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In *SIGMOD Conference*, 1993; 207–216.
26. Lu W, Ghorbani AA, Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal Processing* 2009: article no. 4.
27. Soule A, Salamatian K, Taft N. Combining filtering and statistical methods for anomaly detection. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, 2005.
28. Kephart JO, Chess DM. The Vision of Autonomic Computing. Available: http://users.soe.ucsc.edu/~griss/agent-papers/ieee-autonomic.pdf [last accessed 12 May 2013].
29. IBM. An Architectural Blueprint for Autonomic Computing (2005). Available: http://www.ginkgo-networks.com/IMG/pdf/AC_Blueprint_White_Paper_V7.pdf [last accessed 12 May 2013].
30. Strassner J, Agoulmine N, Lehtihet E. FOCALE: a novel autonomic networking architecture. *ITSSA Journal* 2007; **3**(1): 64–79.
31. Strassner J, Foghl M, Donnelly W, Agoulmine N. Beyond the knowledge plane: an inference plane to support the next generation Internet. In *Proceedings of First International Global Information Infrastructure Symposium (GIIS)*, 2–6 July 2007, Marrakech, Morocco.
32. Schieder A, Abramowicz H, Malmgren G, Sachs J, Horn U, Prehofer C, Karl H. Ambient networks: an architecture for communication networks beyond 3G. *IEEE Wireless Communications* 2004; **11**(2): 14–22.
33. Bouabene G, Jelger C, Tschudin C, Schmid S, Keller A, May M. The autonomic network architecture (ANA). *IEEE Journal on Selected Areas in Communications* 2010; **28**(1): 4–14.
34. GENI (2012). Available: http://www.geni.net/ [26 April 2013].
35. Oates T. Fault identification in Computer Networks: A Review and a New Approach. Technical Report CS-TR 95–113, University of Massachusetts, Amherst, MA, 1995.
36. Chandola V, Banerjee A, Kumar V. Anomaly detection: a survey. *ACM Computing Surveys* 2009; **41**(3): article no. 15.
37. Thottan M. Fault detection in IP networks. PhD dissertation, Rensselaer Polytechnic Institute, Troy, NY, 2000. Under patent with RPI.
38. Barford P, Kline J, Plonka D, Ron A. A signal analysis of network traffic anomalies. In *ACM Internet Measurement Workshop*, November 2002.
39. Baras J, Li H, Mykoniatis G. Integrated, distributed fault management for communication networks. *Technical Report CS-TR 98–10*, University of Maryland, April 1998.
40. Gambhir D, Post M, Frisch I. A framework for adding real-time distributed software fault detection and isolation to SNMP-based systems management. *Journal of Network and Systems Management* 1994; **2**(3): 257–282.
41. Hood CS, Ji C. Proactive network-fault detection [telecommunications]. *IEEE Transactions on Reliability* 1997; **46**(3): 333–341.
42. Jiang Q, Adaikkalavan R, Chakravarthy S. NFM^{i}: an inter-domain network fault management system. In *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, 2005; 1036–1047.
43. Krishnamurthy B, Sen S, Zhang Y, Chen Y. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of ACM Internet Measurement Conference*, October 2003.
44. Ye T, Kalyanaraman S, Harrison D, Sikdar B, Mo B, Kaur HT, Vastola K, Szymanski B. Network management and control using collaborative on-line simulation. In *Proceedings of CNDSMS*, 2000.
45. Papavassiliou S, Pace M, Zawadzki A, Ho L. Implementing enhanced network maintenance for transaction access services: tools and applications. In *Proceedings of IEEE International Conference on Communications*, Vol. 1, 2000; 211–215.
46. Han J, Kamber M. Data Mining Concepts and Techniques (2nd edn). Morgan Kaufmann: Burlington, MA, 2006.
47. Salfner F, Schieschke M, Malek M. Predicting failures of computer systems: a case study for a telecommunication system. *IEEE 20th International Parallel and Distributed Processing Symposium*, IPDPS, April 2006.
48. Salfner F, Malek M. Using hidden semi-Markov models for effective online failure prediction. In *26th IEEE International Symposium on Reliable Distributed Systems*, SRDS, October 2007; 161–174.

DOI: 10.1002/nem

AUTHORS' BIOGRAPHIES

Hesham J. Abed received his B.S. and M.S. degrees from the University of Mustansiriyah, and Western Michigan University, in 1995 and 2011 respectively. He is currently a Ph.D. candidate at Western Michigan University. He is a Computer Science Instructor at the University of Mustansiriyah, Computer Science Department, Baghdad, Iraq, since 1995.

Ala Al-Fuqaha (S'00-M'04-SM'09) received his M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Missouri, in 1999 and 2004 respectively. Currently, he is an Associate Professor and director of NEST Research Lab at the Computer Science Department of Western Michigan University. Before joining Western Michigan University, Dr. Al-Fuqaha was a senior member of technical staff at LAMBDA Optical Systems in Reston, Virginia where he worked on the design and development of embedded routing protocols and network management systems. Prior to LAMBDA, Dr. Al-Fuqaha was a Software Engineer with Sprint Telecommunications Corp. where he worked as part of the architecture team. His research interests include Network Management and Planning, QoS routing in optical and wireless networks, and performance analysis and evaluation of high-speed computer and telecommunication networks. Journal. Dr. Al-Fuqaha is a senior member of the IEEE and has served as Technical Program Committee member and reviewer of many international conferences and journals.

Bilal Khan is Professor of Mathematics and Computer Science at John Jay College, City University of New York (CUNY) where he is doctoral faculty in the programs of Computer Science, Forensic Computing, and Criminal Justice. He received his B.Sc. from MIT (1993), M.Sc. from John Hopkins (1997) and Ph.D from CUNY (2003). He is the author of over 75 refereed journal and conference publications on theoretical and empirical aspects of networks.

Ammar Rayes, Ph.D., is a Distinguished Service Engineer at Cisco Systems leading Smart Services technology strategy including Internet of Everything (IoE), Machine to Machine, Big Data and analytics. He is also the president of The International Society of Service Innovation Professionals. He has authored / co-authored over a hundred papers and patents on advances in telecommunications-related technologies, including a book on Network Modeling and Simulation and one on ATM switching and network design. He is an Associate Editor of ACM "Transactions on Internet Technology", Editor-in-Chief for "Advances of Internet of Things" Journal. He has also served as an Advisory Board Member of the Journal of Wireless Communications and Mobile Computing. Ammar chairs Cisco's Smart Services Patent Council and the University Research Program for Smart Services. He received his BS and MS Degrees in Electrical Engineering from the University of Illinois at Urbana in 1986 and 1988, respectively. He received his Doctor of Science degree in Electrical Engineering from Washington University in St. Louis, Missouri, in 1994 where he received the Outstanding Graduate Student Award in Telecommunications.

DOI: 10.1002/nem