A Hybrid Classifier Using Reduced Signatures for Automated Soft-Failure Diagnosis in Network End-User Devices

C. Widanapathirana X. Ang J. C. Li M. V. Ivanovich P. G. Fitzpatrick Y. A. Şekercioğlu
Department of Electrical and Computer Systems Engineering, Monash University, Australia
{chathuranga.widanapathirana, xavier.ang, jonathan.li, milosh.ivanovich, paul.fitzpatrick, ahmet.sekercioglu}@monash.edu

***Abstract*— We present an automated system for the diagnosis of both known and unknown soft-failures in end-user devices (UDs). Known faults that cause network performance degradation are used to train the classifier-based system in a supervised manner while unknown faults are automatically detected and clustered to identify the existence of new categories of soft-failures. The supervised classifier used in the system can be retrained by including the newly detected faults to enhance its performance.**

The system uses 460 features to construct Normalized Statistical Signatures (NSSs) for fault characterization. Due to the high dimensionality of NSSs, EigenNSS was proposed to reduce the complexity without losing important information. Because of the natural network inconsistencies that exist in communication links, we propose FisherNSS, a reduced signature that provides improved linear separability between classes to further enhance classification performance.

The system is evaluated over a live campus network using 17 emulated UD faults. The results show that the best overall classification accuracy of up to 97% was achieved by using FisherNSS with a dimensionality reduction of 96.74%. In comparison, both EigenNSS and FisherNSS have faster training and diagnosis time compared to NSS, which makes them suitable for on-demand as well as real time diagnostic applications. Furthermore, FisherNSS compared to EigenNSS has a higher diagnostic accuracy and quicker diagnosis time (order of microseconds).

I. Introduction

Network performance problems affect many end-users, ranging from everyday internet users to large corporations. Studies have shown that these problems are caused by service provider servers, backbone networks, access networks, or the end-user devices (UDs) themselves [1]. The performance of a network is usually defined by two main categories [2]: *hard-failures* and *soft-failures*. Hard failures correspond to the inability to transfer any data between users and the network, which can be easily identified and resolved due to immediately noticeable loss in connectivity. Soft-failures are characterized by degraded performance, which are more difficult to diagnose and are usually assisted by Network Monitoring Systems (NMS) to collect signs from the network. However, interpreting such signs to diagnose the root causes of the problem still require expensive resources which include intervention by skilled personnel.

Possible root causes of soft-failures in UDs are

- misconfiguration of parameters in the protocol layers, generally due to the conservative default values in operating systems [3],
- Hardware problems such as new application installations, NIC driver issues,
- kernel level software problems,
- mismatch between system settings and the link [4], or
- protocol implementation errors [5].

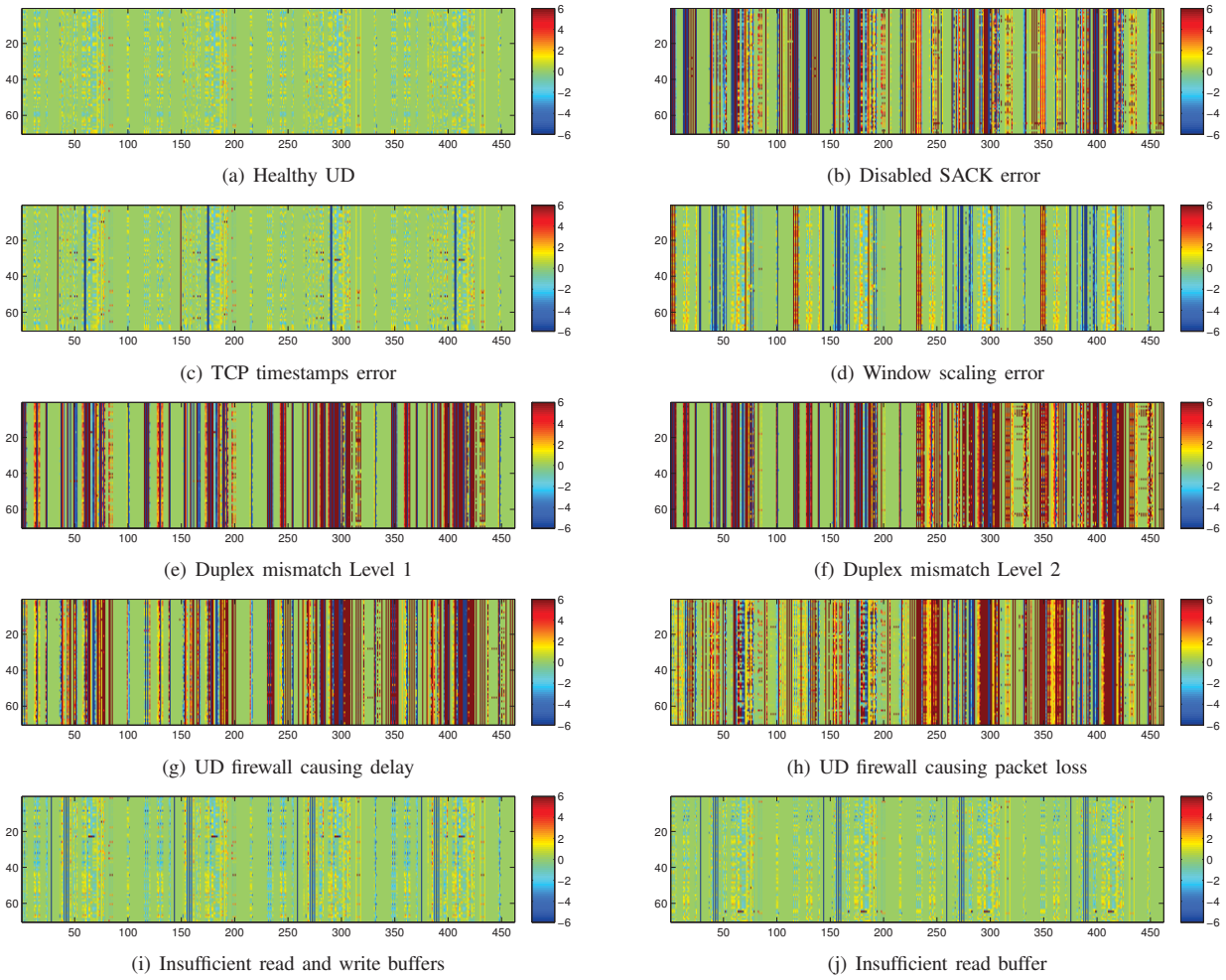
Recently, researchers have proposed an automated diagnosis solution especially focused on core networks, access networks and servers [6] but there is not much development taking place in automated solutions for UDs. Diagnosis methods based on collected packet traces over a TCP (Transmission Control Protocol) connection have been shown to be effective for finding the root causes of network performance problems [7]. Collected packet traces contain artifacts that represent behavioural characteristics of the network. These characteristics can be utilized by skilled investigators to identify the root causes of the faults. Another advantage of trace analysis-based diagnosis approach is that the traces can be collected very easily without the requirement of any special equipment.

A. Motivation

Our search on the research literature has revealed the lack of a fully automated solution for identifying the root causes of network soft-failures using TCP traces. To fill this gap, we previously proposed an automated diagnostic system based on supervised Machine Learning (ML) algorithms and network fault signatures[1] created using aggregated TCP statistics [8], [9]. In our system, the ML algorithms are first trained using signatures that we call as *Normalized Statistical Signatures (NSS)*, which are generated from collected packet traces. However, the diagnostic capability of the system is limited by the number of fault classes present in the training set. In the

[1] A *signature* is often defined as a collection of features, where each of these features represents a single aspect of behavioural characteristics in the network.

doi:10.4304/jnw.9.12.3275-3289



(a) Healthy UD

(b) Disabled SACK error

(c) TCP timestamps error

(d) Window scaling error

(e) Duplex mismatch Level 1

(f) Duplex mismatch Level 2

(g) UD firewall causing delay

(h) UD firewall causing packet loss

(i) Insufficient read and write buffers

(j) Insufficient read buffer

Figure 1. Visualization of NSSs for a healthy UD and nine common UD soft-failures. Here, the columns of each figure represent 460 features of the 70 NSSs. Each NSS occupies a row. Features have been normalized and scaled to $[-6, 6] \subset \mathbb{R}$ and their values are represented by colored pixels to project the scaled feature values to RGB space. This representation offers easy visualization and comparison. The images demonstrate that the combination of features uniquely represents each fault and can be used as a "fingerprint" for diagnosis.

case of diagnosing an unknown fault, this often leads to a false positive error. We have also found that the NSSs contain large numbers of features, which could cause over fitting of the classifier models, a problem known as the "curse of dimensionality" [10].

As a step to compensate for the limitations encountered, we present a new hybrid classifier architecture that extends the diagnosis capability of the system to both previously known faults as well as new types of faults. The hybrid classifier system combines unsupervised clustering algorithms[11], [12] to analyze previously unknown signatures to detect new faults and iterative training of a supervised ML classifier for root-cause diagnosis.

We also present two new signatures called *EigenNSS* and *FisherNSS*, both motivated by techniques used in facial recognition applications. The new signatures transform the NSSs to lower dimensions without sacrificing useful information. In addition, FisherNSS strives to maximize the ratio of the between-class scatter to the within-class scatter for better classification results. We perform a detailed comparison of performance between both EigenNSS and FisherNSS with data gathered from real-world networks. Preliminary results of the work have been published in a conference paper [13]. Whilst the published work only offer a limited discussion focused just on EigenNSS, this publication significantly extends the concept to introduce the FisherNSS. Additionally, this publication offer much detailed discussion and performance evaluation on both types of transformed signatures as well the hybrid classifier system.

II. System Overview

A. Normalized Statistical Signature (NSS)

In our work, we use a self-initiated controlled connection between the diagnostic server and UD to collect TCP packet traces. The collected traces are sent through a feature extraction module. In this paper, we use 230 extracted features from each trace, which totals to 460 when both upload and download traces are combined. Features extracted include cumulative totals of packet types, payload characteristics, observation frequencies of specific events, initial and final state parameters, delay

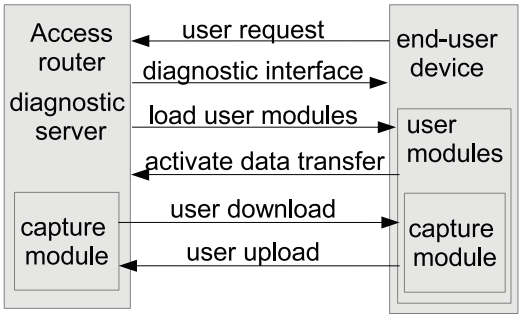


Figure 2. Deployment of the diagnostic system over the access network and operational overview.

based statistics, and TCP Boolean parameters. Various other extracted features that was included can be found in [14].

To standardize all the signatures, each of the 'faulty' raw signatures are normalized against the healthy baseline signature, which is obtained using a UD with optimal system settings. The resultant feature vector is called the Normalized Statistical Signature (NSS). Figure 1 shows NSSs collected from UDs exhibiting 9 types (classes) of faults and one 'healthy'[2] UD. Throughout all 70 samples for each type of fault, unique feature patterns are observed, providing the distinction needed for an effective and reliable diagnosis.

Since an NSS contains a large feature set (460 features), it theoretically enables a range of faults to be characterized through a single representation. However, as evident from the Figure 1, not all features contribute equally for the separability of the classes and even features within the same class show variations due to the inconsistent nature of the connection link. Having a large feature set can also lead to over fitting of the data when training a ML-based system; this consequently will lead to poor generalization and classification accuracy. Therefore, the dimensionality of the NSSs should be reduced while preserving the important information to build an effective diagnostic system.

In this study, we include two techniques to achieve dimensionality reduction in NSSs. Principal Component Analysis (PCA) is used to transform the NSS into a new signature called EigenNSS whereas Fisher's Linear Discriminant Analysis (FLDA) is used to generate another signature type called FisherNSS [15].

B. Operational Details of the Diagnostic System

1) Deployment: The diagnostic server is deployed as an application on the access router as shown in Figure 2. The complexities that can affect the uniformity of the captured packet traces can be eliminated by narrowing down the path to the UD into an access link. Firstly, upon initiation from the user, the modules needed for file transfers and packet captures are loaded. Two TCP-based data transfers of a fixed size of 20 MB file (an upload and download) are conducted serially between the UD and server. A file size of 20 MB was found, empirically, to provide the best balance between signature accuracy and collection time. A self-contained, portable packet-capturing mechanism that does not require kernel manipulations or installations is used to capture the packets. To satisfy the privacy concerns, we limit the amount of traffic captured and also control the content to be captured. Finally, the captured TCP packet traces are sent to the feature extraction modules where they are analysed and extracted to obtain statistical attributes (features), known as 'raw' signatures.

[2]In this work, we define a 'healthy' UD as a device that receives the maximum network performance level reasonably expected by a user.

D	Selected number of eigenvectors that account for the highest variation
p	Number of samples in signature set
m	Number of features in each NSS (m = 460)
$\mathbf{x}$	m-dimensional feature vector of the training NSS set
Φ	Mean of the training NSS set
Δ	Mean centered training NSS set
Ψ_{PCA}	Eigen matrix formed using a pre-determined D number of eigenvectors, v
Θ_{PCA}	EigenNSS is generated by projecting the mean centered NSS onto the Eigen matrix (Θ_{PCA} = $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p\}$)
Φ_{PCA}	Mean of the entire EigenNSS set
Φ_i	Mean of the EigenNSS for the i^{th}
S_B	Between-class scatter
S_W	Within-class scatter
Ψ_{FLD}	Eigen matrix formed using a pre-determined D number of eigenvectors, w
Θ_{FLD}	FisherNSS is generated by projecting EigenNSS onto the Fisher matrix (Θ_{FLD} = $\{\mathbf{f}_1, \mathbf{f}2, \ldots, \mathbf{f}_p\}$)
$\epsilon_{f,\text{PCA}}$, $\epsilon_{f,\text{FLD}}$	Pattern vectors of each class
$\sigma_{f,\text{PCA}}$, $\sigma_{f,\text{FLD}}$	Euclidean distance between the test sample with each of the class pattern vectors
μ	Squared distance between the test NSS feature vector sample and the mean of the training NSS set

TABLE I.
MATHEMATICAL NOTATION

2) Operation: Figure 3 shows the operational stages of the diagnostic system. There are three main stages that the system operates in

1) Training stage
2) Diagnosis stage which also includes the
3) New class recognition stage.

During the training phase, the packet traces are colu-vjlected from UDs with known faults induced. Statistical attributes of these traces are extracted to form the *raw signature* and given a class label. The raw signatures are then normalized against the healthy performance baseline to create the NSSs. The NSSs generated from multiple classes are stored in a database and are used to calculate the *transformation matrices* of the database. The NSSs are projected onto either one of the transformation matrices to create the transformed signatures, EigenNSS and FisherNSS respectively. The transformed signatures of each class are used to calculate the pattern vector (ϵ_f) (Table I can be referred for the descriptions of the mathematical symbols used throughout the paper) of that

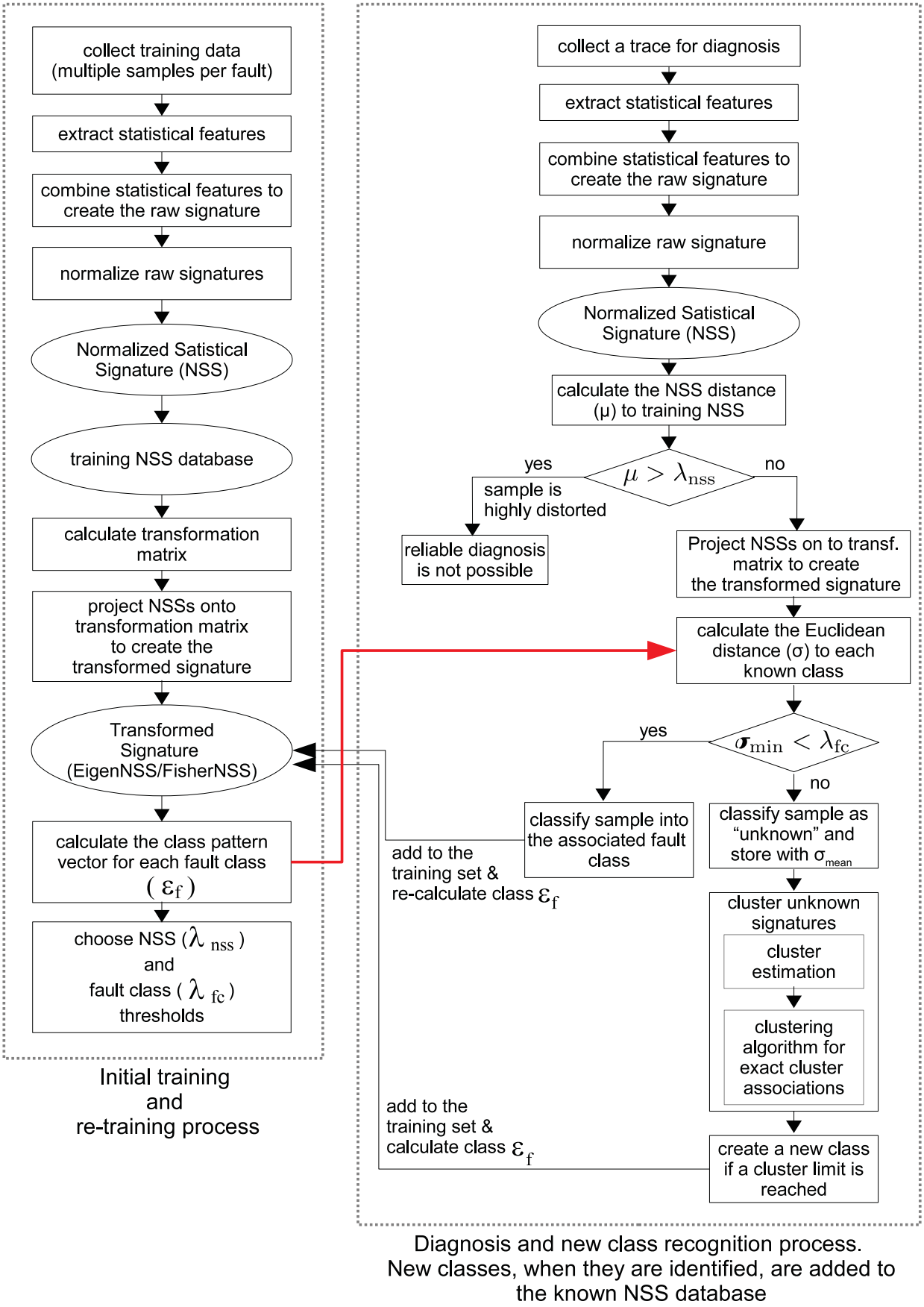


Figure 3. Operational overview of the classifier system.

particular class. Finally, two thresholds are chosen for the system:

1) λ_{nss} for determining if a particular signature is valid, and
2) λ_{fc} for determining class associations.

Once the system is trained, UDs that are suspected to be faulty can be diagnosed by collecting packet traces and sending them through the same feature extraction and NSS generation process. Then, EigenNSS and FisherNSS can be generated by projecting the NSSs onto either one of the transformation matrices created during the training stage. These transformed signatures are then used to calculate the pattern vector (ϵ) and Euclidean distance (σ) of the pattern vector from each known class. The NSS can also be used to calculate the square distance (μ) from the training NSS data set. Finally, depending on the minimum Euclidean distance ($\sigma_{\min}$) and square distance (μ), one of the three possible outcomes is decided by the system as follows

if $\mu > \lambda_{\text{nss}}$ **then**
  The sample is highly distorted.
  Reliable diagnosis is not possible.
else
  if $\sigma_{\min} < \lambda_{\text{fc}}$ **then**
    The sample is classified to the class associated with the minimum distance. The sample then is added to the training NSS database and the ϵ_f of the class is recalculated to include the new sample.
  else
    The sample contains a valid signature, but it does not belong to any of the known classes. Hence, the sample is classified as an unknown class.
  end if
end if

When the system detects a predetermined number of unknown signatures, new class recognition phase begins. This predetermined number is usually defined to be twice the minimum threshold size that a given cluster is accepted as a new class. These unknown signatures are first stored in a separate database with their pattern vectors. Then, these signatures are sent through a cluster estimation algorithm [11] which determines (i) if the data set has samples that can be clustered within the λ_{fc} bound, and (ii) the number of clusters that can be created. These clusters will be matched with their exact cluster memberships with the assistance of a clustering algorithm, which uses a fuzzy C-means clustering technique with iterative optimization [12]. If any of the clusters reach the minimum threshold size, they will be considered as a new class and will be added into the training database. The transformed signatures of the new class are sent to the class pattern vector calculation while its NSSs are sent to the classifier training database. Although new classes can be added by calculating the class pattern vectors, the system can be re-trained in a short time when it isn't performing any diagnostics if the training database is updated with the new NSSs. Re-training includes the new class and improves the final accuracy of the system. The system also prompts administrators that a new fault class has been detected, and after investigative analysis of its actual root cause, a class label can be created.

The rest of the paper is organized as follows. Section III recaps some of the related work done by other researchers. Section IV presents how the NSSs are transformed to generate EigenNSS and in Section V, the creation of the new signature, FisherNSS is discussed elaborately. Section VI includes the training and diagnosis process of the system. Finally, Section VII contains detailed performance analyses of the systems comparing both types of signatures used followed by the conclusion in Section VIII.

III. Related Work

Characterizing the behaviour of the network to model a signature is an approach found mainly in network applications with detection tasks. A signature is defined as a collection of features (or attributes), each representing a single aspect of the network's behaviour. In a detection process, signatures provide the key to differentiate "healthy" behaviours from the abnormal or faulty ones. However, depending on a particular application, the generation process of a signature may vary to account for the requirements and constraints. In order to find a suitable network signature, investigations about several available types of network signatures are carried out to maximize the capability of our diagnostic system. In addition to that, various existing dimensionality reduction techniques are reviewed, weighing between their pros and cons.

A. Network signatures

This subsection shows a summary of the different types of network signatures and their limitations in the context of UD, soft-failure characterization. Firstly, network signatures generated from flow-based characteristics have been commonly used in online traffic classification as in Roughan *et al.* [16] and IDSs as in Zhang *et al.* [17]. Kihara et al. [18], Hajji [19] and Thotta and Ji [6] have used signatures created using the behavioral changes in traffic flows for network fault detection. These applications focus on detecting abnormalities in the overall traffic flow pattern of the network when compared with the normal traffic flow. However, these flow-based signatures are not suitable for UD diagnosis applications due to their passive and continuous monitoring nature. Our application requires diagnosis to be run on-demand when a user experiences a network performance problem.

Another common approach is to use system logs from devices or 'reports' compiled by the user. Aggarwal *et al.* [20] and Reidemeister *et al.* [21] incorporated internal system logs whereas Lee and Kim [22] used user-reports to create their respective fault signatures. The usage of the system logs not only brings up privacy concerns in public networks but is also inconvenient for network operators as

they have to gain privileged access to the UDs. Reports generated by users may be unreliable if users have no specific network knowledge. Although system logs and user-reports can provide valuable information when generating a fault signature, we believe the challenges far outweigh the benefits.

Communication protocols are another common source of information to create network signatures. Popular protocols such as IP, TCP, UDP, and HTTP have often been used because they are usually supported by most devices. Dahmouni *et al.* [23], Manikopoulos and Papavassiliou [24], and Wolfgang [25] extracted features from multiple protocols, whereas other studies such as Gomes *et al.* [26] and Chen *et al.* [27] have limited feature collection to a single protocol. However, these proposed signatures have been created to detect a very specific network problem which doesn't provide us with much flexibility for our application. In the case of UD soft-failure detection, the requirements are to be able to effectively characterize not only a large number of faults, but also any new types that are unknown to our system. Hence, these proposed signatures with limited features can limit the ability to capture valuable information needed to generalize the signatures.

Our literature review has revealed that the existing network signatures are unsuitable for our application as they do not offer satisfactory solutions to characterize UD soft-failures. Therefore, we propose a more comprehensive signature to characterize UD soft-failures and analyse how uniquely different the signatures are on different network properties.

B. Complexity reduction

This subsection shows a summary of the different types of dimensionality reduction techniques used by other researchers and their limitations. Network signatures generated from flow-based characteristics for traffic classification such as the one illustrated by Zhang *et. al.* [17] contain large amounts of data as they are collected from one of China's seven major backbone networks. The complexity of their dataset was reduced by only capturing packet headers, which surprisingly still contain excessive amount of data for a whole day. For further complexity reduction, each day is divided into 24 hours where data is only captured in the first minute for each hour. Due to the on-demand requirement, this reduction method is deemed unsuitable for our application.

Clustering algorithms are commonly used in reducing the complexity of a large data set. Vaarandi proposed a density-based approach clustering [28] to reduce the amount of data (signatures from system log files) required by a support person to evaluate the behaviour of the system. In a density-based clustering, clusters are usually defined as areas of higher density than the remaining data set. The algorithm consist of three steps which include (i) data summarization, (ii) building cluster candidates, using the summary information collected and finally, (iii) cluster selection based on the candidates. This method not only clusters data into regions based on frequent patterns from the log files but also extracts the static parameters that are unique to the system.

In other domains, Fisher's Linear Discriminant (FLD) [15] is used to reduce the dimensionality of image data sets. This method is also very versatile in overcoming inconsistencies and variances in an image (eg. lighting variations in facial recognition). FLD provides linear class separation in huge data sets which helps simplify classification process.

Our complexity reduction method was greatly motivated by how well facial recognition methods have performed in their respective fields and we were inspired to try these methods on our generated signatures.

IV. EigenNSS: NSS Transformation Using Principal Components

To overcome the challenges with having high dimensionality in NSSs, we searched for a way to emphasize on significant global features that contain a maximum amount of information. Dimensionality reduction can be achieved by firstly finding the principle components (i.e eigenvectors of the covariance matrix) of the distribution of NSSs. Then, these eigenvectors can be ordered according to the amount of variation among the NSSs. Finally, the NSSs are projected onto a selected number (D) of eigenvectors that accounts for the highest variation. These projections are called "EigenNSS" which represent most of the information from the original samples in a D-dimensional space.

A. Generating EigenNSS

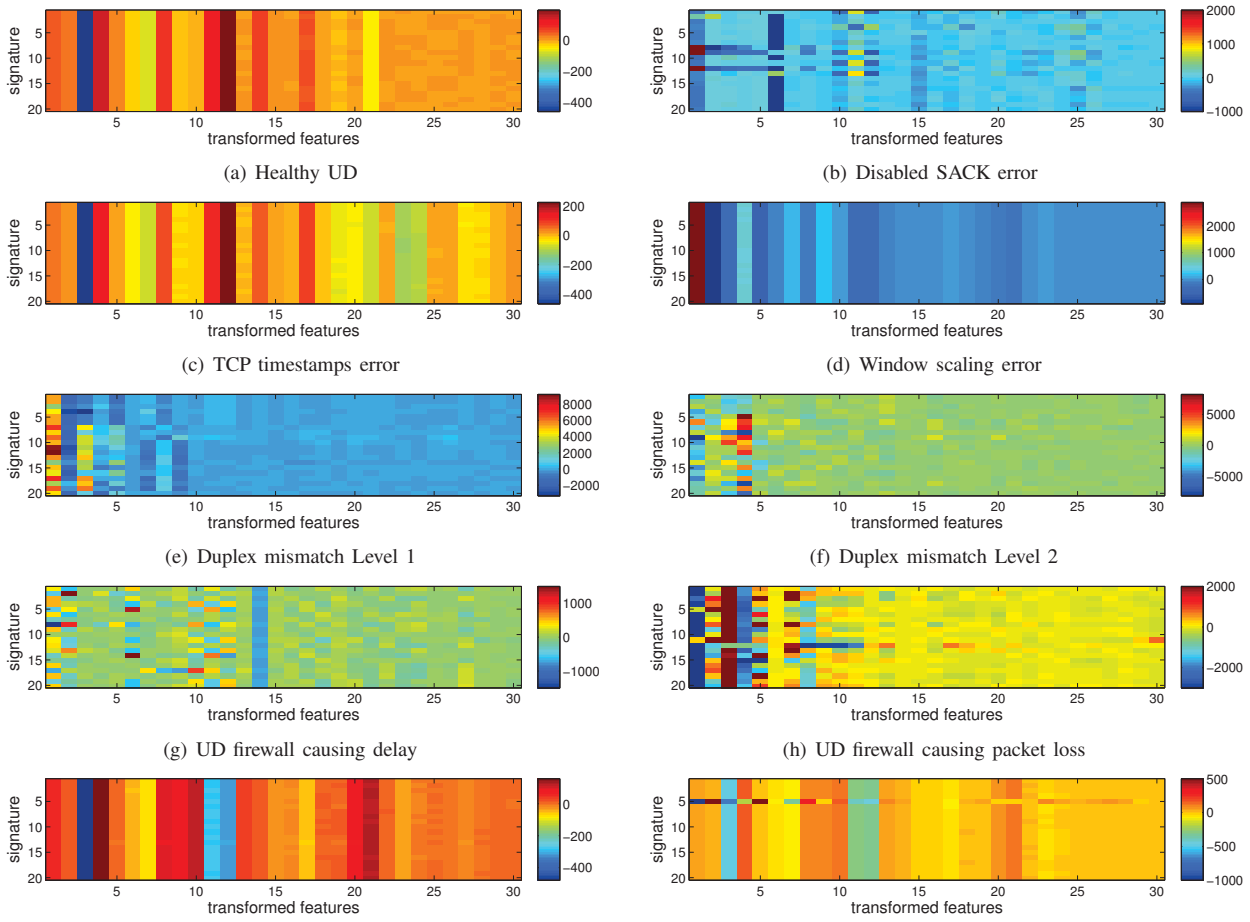
The following subsection provides details of the EigenNSS generation and calculation process. Consider a training NSS set of $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_p$ where x is a m-dimensional feature vector. For example, the set of NSS shown in Figure 1 has 70 samples for each of the 10 classes (p = 70 x 10 = 700), in which each of them has 460 feature vector (m = 460). Principal Component Analysis (PCA) can be performed either on co-variance matrix or on correlated matrix and the choice usually depends on the variance of features. Correlation matrix is preferred when the scales of the features are significantly different from one another. When the scales of the features are similar, the covariance matrix is preferred, as the correlation matrix will lose information when standardizing the variance. We have chosen to perform PCA on covariance matrix since features in NSSs are already normalized and approximately have similar scales.

The mean of the NSSs can be found by

$$\Phi = \frac{1}{p}\sum_{k=1}^{p}\mathbf{x}_k$$

The training NSSs are then mean centered by

$$\bar{\mathbf{x}}_i = \mathbf{x}_i - \Phi$$



(a) Healthy UD (b) Disabled SACK error

(c) TCP timestamps error (d) Window scaling error

(e) Duplex mismatch Level 1 (f) Duplex mismatch Level 2

(g) UD firewall causing delay (h) UD firewall causing packet loss

(i) Simultaneous insufficient read-write buffer (j) Insufficient read buffer

Figure 4. Comparison of EigenNSSs for common UD soft-failures.

where $\bar{\mathbf{x}}_i$ is the mean centered m-dimensional feature vector of the i^{th} instance. The resultant matrix $\Delta = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, ..., \bar{\mathbf{x}}_p\}$ has the dimensions of m x n and used to calculate covariance matrix Δ_{cov}. Δ_{cov} is then subjected to PCA where eigenvectors, $\mathbf{v}_i$ and the corresponding eigenvalues n_i are determined by solving a well-known singular value decomposition (SVD) problem. Then, a pre-determined D number of eigenvectors, v that are arranged from the highest eigenvalue, n are selected to create the Eigen matrix which has the dimensions of m x D. Finally, EigenNSSs are created by projecting the mean centered NSS matrix, Δ onto the Eigen matrix, Ψ_{PCA} as

$$\Theta_{\text{PCA}} = {\Psi_{\text{PCA}}}^T \Delta$$

where $\Theta_{\text{PCA}} = \{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_p\}$ and $\mathbf{e}$ is a D-dimensional EigenNSS.

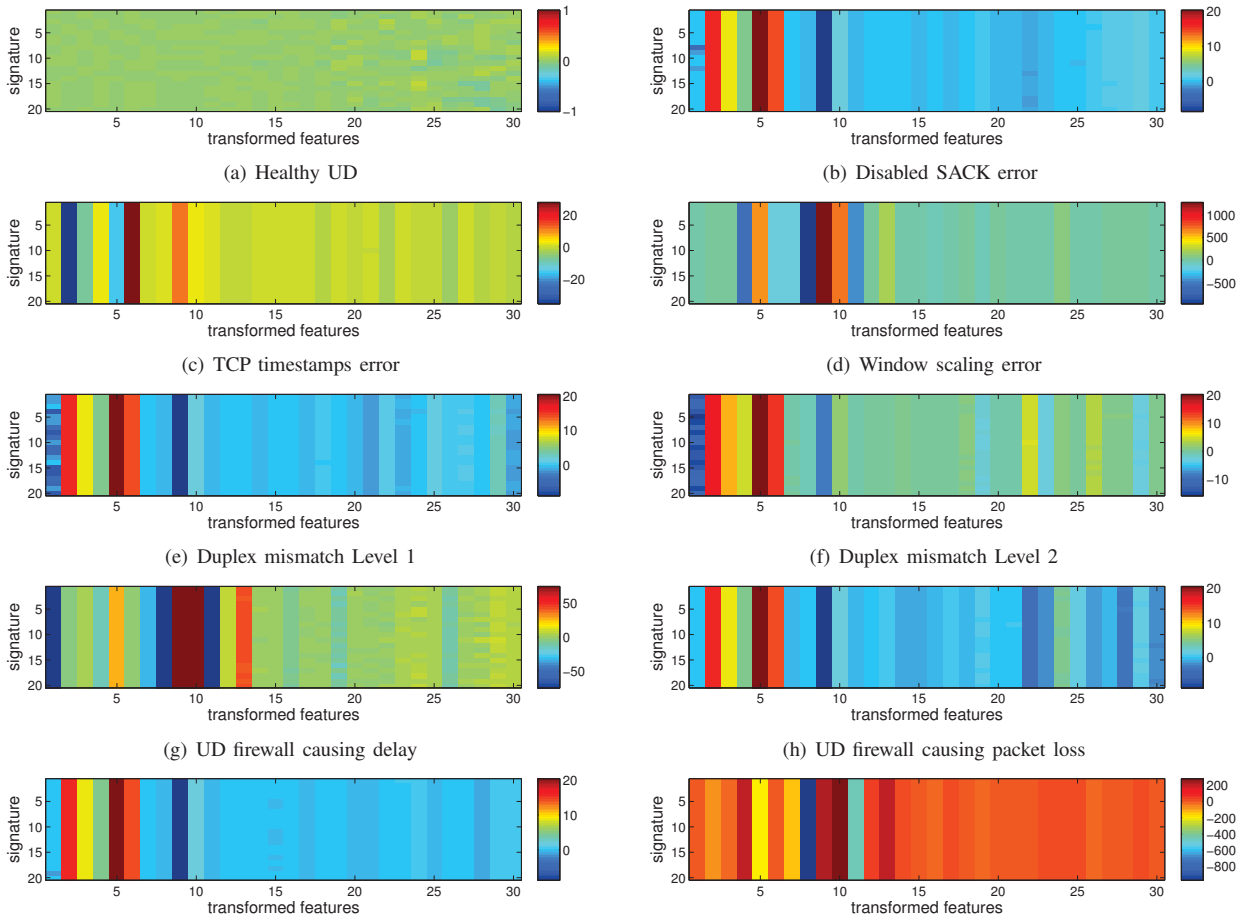
Figure 4 shows the comparison of EigenNSSs for various types of common UD faults with D=30. Note that the figure only shows 25 signature samples per class for clarity. As shown in the figure, the dimensionality of NSSs has been reduced while preserving the most important information for class separation. This is clearly visible as the EigenNSS shows significant differences between classes compared to the NSSs. However, it can be seen that the EigenNSS samples within the same classes can vary due to the inconsistent nature of the network links.

V. FisherNSS: NSS Transformation Using Fisher's Linear Discriminant Analysis

As shown in the previous section, EigenNSS reduced the dimensionality of NSS. However, to account for the errors in data collection and the inconsistent nature of the networks, separation between these unwanted information is needed for a better classification outcome. This section shows the motivation and generation process of FisherNSS, a new transformed signature.

A. Fisher's Linear Discriminant

The inconsistent nature of the network affects the extracted features and may lead to a poor fault detection. These inconsistencies (noise terms) are embedded inside the data which makes it difficult to distinguish from the actual information. PCA used in EigenNSS calculates the eigenvalues that explain most of the variation across the data; in this case it would operate per feature vector and does not take account of class labels. As previously mentioned, Fisher's Linear Discriminant (FLD) aims to maximize Fishers discriminant ratio, i.e. it maximizes the



(a) Healthy UD

(b) Disabled SACK error

(c) TCP timestamps error

(d) Window scaling error

(e) Duplex mismatch Level 1

(f) Duplex mismatch Level 2

(g) UD firewall causing delay

(h) UD firewall causing packet loss

(i) Simultaneous insufficient read-write buffer

(j) Insufficient read buffer

Figure 5. Comparison of FisherNSSs for common UD soft-failures.

distance between classes and provides linear separability between classes, to facilitate correct fault diagnosis. However, to guarantee within class scatter matrix in FLD not to become singular, we require at least $x + c$ samples (x=number of dimensions, c=classes) and in the case of NSS, at least 475 samples per class. This requirement reduces the usability of FishersNSS to more mature systems with large data sets. To reduce the minimum sample requirement, we use a well-established two-phase framework of PCA plus FLD where PCA first reduces the dimensions of the feature space, and then apply Fisher's Linear Discriminant Analysis (FLD) for further reduction and between class separation [29], [30].

FLD is an example of a *class specific method*, that attempts to "shape" the scatter of feature values to facilitate reliable classification. To illustrate the benefits of a class specific linear projection, we construct a set of 10 2-dimensional (n=2) sample points. In Figure 6, a comparison of both PCA and FLD for a two-class problem is shown by projecting the constructed sample points from 2D down to 1D respectively. Comparing the projections, PCA smears the classes together so that they are no longer linearly separable in the projected space. Although the total scatter of FLD is smaller than of

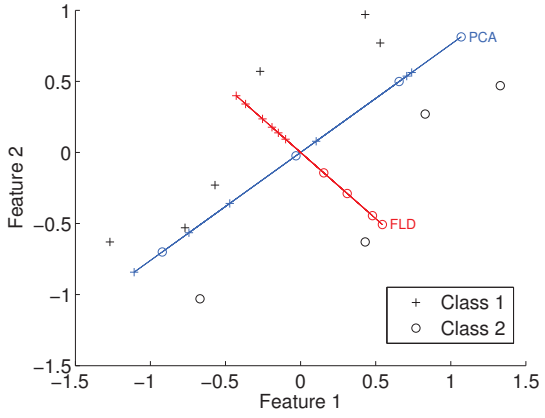


Figure 6. A comparison of principal component analysis (PCA) and Fisher's Linear Discriminant (FLD) for a two class problem.

PCA, FLD achieves greater between-class scatter, and consequently results in better classification.

B. Generating FisherNSS

The following subsection illustrates how the FisherNSS is generated including the calculation process. Assuming we have a training EigenNSS set, Θ_{PCA} of $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_p$

, where **e** is a D-dimensional transformed feature vector. For example, the set of EigenNSS shown in Figure 4 has N=25 samples for each of the 10 classes (p = 25 x 10 = 250), in which each of them has D=30 features. The mean of the entire EigenNSS set can be found by

$$\Phi_{\text{PCA}} = \frac{1}{p}\sum_{k=1}^{p}\mathbf{e}_k$$

The mean of the EigenNSS for the i^{th} class can be found by

$$\Phi_i = \frac{1}{N_i}\sum_{\mathbf{e}_k \in E_i}\mathbf{e}_k$$

where Φ_i is the mean EigenNSS of class E_i and N_i is the number of samples in class E_i. The between-class scatter matrix can then be computed by

$$S_B = \sum_{i=1}^{c}(\Phi_i - \Phi_{\text{PCA}})(\Phi_i - \Phi_{\text{PCA}})^T$$

and the within-class scatter matrix by

$$S_W = \sum_{i=1}^{c}\sum_{\mathbf{e}_k \in E_i}(\mathbf{e}_k - \Phi_i)(\mathbf{e}_k - \Phi_i)^T$$

The optimal projection W_{opt} is chosen as the matrix with orthonormal columns (eg. eigenvectors, w) which maximizes the ratio of the determinant of the between-class scatter matrix to the determinant of the within-class scatter matrix, i.e.,

$$W_{\text{opt}} = \arg\max\frac{|W^T S_B W|}{|W^T S_W W|}$$

However, only a pre-determined D number of eigenvectors that are arranged from the highest eigenvalue, are selected to create the Fisher matrix which has the dimensions of D x D. Finally, FisherNSS are created by projecting the EigenNSS matrix, Θ_{PCA} onto the Fisher matrix, Ψ_{FLD} as

$$\Theta_{\text{FLD}} = {\Psi_{\text{FLD}}}^T\Theta_{\text{PCA}}$$

where $\Theta_{\text{FLD}} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_p\}$ and **f** is an D-dimensional FisherNSS.

Figure 5 shows the FisherNSSs for various types of common UD faults with D=30. Similarly, only 25 signature samples per class are shown for clarity. As evident from the figure, the signature samples in each class exhibit very little variation and appear to reduce the effects of network inconsistencies on the extracted signatures.

VI. Training and Diagnosis of Reduced Signatures

A. Training

The following subsection shows the diagnostic system training process using EigenNSS and FisherNSS respectively, where the pattern vectors (ϵ_f) of each class are calculated. We assume the EigenNSS matrix, Θ_{PCA} and FisherNSS matrix, Θ_{FLD} contains n signature samples from multiple classes (faults, f). The D-dimensional class pattern vector, ϵ_f is calculated by averaging the reduced signatures as

$$\epsilon_{f,\text{PCA}} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{e}_k$$

$$\epsilon_{f,\text{FLD}} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{f}_k$$

where these pattern vectors, ϵ_f are used during the diagnosis stage to calculate the distance between any given unknown (test) signatures. These distances are used to determine the best class fit of the unknown signatures.

B. Diagnosis

1) Known Faults: In this subsection, the classification criteria required for the diagnosis of known faults are shown. The simplest method of determining the class association of a test sample is to calculate the Euclidean distance, σ between **e** and **f** respectively with each of the class pattern vectors as

$$\sigma_{f,\text{PCA}} = ||\mathbf{e} - \epsilon_{f,\text{PCA}}||^2$$

$$\sigma_{f,\text{FLD}} = ||\mathbf{f} - \epsilon_{f,\text{FLD}}||^2$$

Euclidean distance assumes the data to be isotropically Gaussian. Euclidean distance was used over more computationally expensive measures such as Mahalanobis distance because our previous analysis of NSSs have shown that features are largely uncorrelated and clearly follow a Gaussian distribution[31]. The test sample is classified and associated with a class, f when its respective minimum Euclidean distance, σ_{min} is below the chosen fault class threshold, λ_{fc}.

Due to errors in data collection and the inconsistent nature of networks, some of the collected packet traces can be distorted. This may lead to a false detection, where the erroneous NSSs generated from these distorted packet traces are wrongly classified. Hence, we introduced another term for a more reliable outcome which is the squared distance, μ between the NSS feature vector of the test sample, y and the mean of the training NSSs, Φ.

$$\mu = ||y - \Phi||^2$$

The test sample is considered valid only if the minimum squared distance is less than the chosen NSS threshold, λ_{nss}.

Depending on the minimum values of σ_f and μ, the outcome of the diagnosis process is determined following the criteria previously mentioned in Section I.

2) Unknown faults: Here, we present the new class recognition process for the diagnosis of unknown faults. A test signature samples is classified as "unknown" if it contains a valid signature but does not belong to any of the known classes. These unknown samples are sent through a cluster estimation algorithm to determine the number of clusters that can be created and whether or not, the samples can be clustered within the λ_{fc} bound.

Assume that we have a set of n unknown signature samples $\{x_1, x_2, ..., x_n\}$ in the database. We consider

each of the unknown samples as a potential cluster center and its respective measured potential, P_i as a function of its distances to all the other unknown samples. The measure of potential for the i^{th} unknown samples are given as

$$P_i = \sum_{j=1}^{n} \mathbf{e}^{-\alpha||x_i - x_j||^2}$$

where

$$\alpha = \frac{4}{r_a^2}$$

and r_a is a positive constant, corresponding to the radius defining a neighbourhood, where unknown samples outside this radius have limited influence on the potential. After the potential of every unknown sample has been computed, we choose the unknown sample with the highest measured potential as the first cluster center.

Let x_1^* be the location of the first cluster center and P_1^* be its measured potential value. The potential of each unknown sample x_i is then revised by subtracting an amount of potential as a function of its distance from the first cluster center. The revised potential of the i^{th} unknown sample can be calculated as

$$P_i = P_i - P_1^* \mathbf{e}^{-\beta||x_i - x_1^*||^2}$$

where

$$\beta = \frac{4}{r_b^2}$$

and r_b is a positive constant, corresponding to the radius defining the neighbourhood that will have measureable reductions in potential. The constant r_b is set to be somewhat greater than r_a, to avoid cluster centers being too closely spaced together. A good choice of values is $r_b = 1.5r_a$. The unknown samples near the first cluster center will have greatly reduced potential, and are unlikely to be the source of the next cluster center. Therefore, the unknown sample with the highest remaining potential is selected to be the second cluster center.

In general, the process of acquiring new cluster centers, x_k^* is repeated using the general formula for revising potentials as

$$P_i = P_i - P_k^* \mathbf{e}^{-||x_i - x_k^*||^2}$$

following these criteria:

if $P_k^* > \overline{\epsilon} P_1^*$ **then**
  Accept x_k^* as a cluster center and continue.
else if $P_k^* < \underline{\epsilon} P_1^*$ **then**
  Reject x_k^* and end process.
else
  Let $d_{\text{min}} \leftarrow$ shortest distances between x_k^* and all previously found cluster centers.
  if $\frac{d_{\text{min}}}{r_a} + \frac{P_k^*}{P_1^*} \geq 1$ **then**
    Accept x_k^* as a cluster center and continue.
  else
    Reject x_k^* and set the potential at x_k^* to 0.
    Select the unknown sample with the next highest potential as the new x_k^* and re-test.
  end if
end if

Here $\overline{\epsilon}$ represents the threshold for the potential above which we will definitely accept the unknown samples as a cluster centers. Whereas $\underline{\epsilon}$ represents a threshold below which we will definitely reject the unknown samples.

Once the cluster data are obtained, they are sent to a clustering algorithm to determine the exact cluster membership. This algorithm uses Fuzzy C-Means (FCM) clustering with iterative optimization that minimizes the cost function

$$J = \sum_{k=1}^{n} \sum_{i=1}^{c} \mu_{ik}^m ||x_k - v_i||^2$$

where n is the number of unknown test samples, c is the number of clusters (obtained from cluster estimation), x_k is the k^{th} unknown sample, c_i is the i^{th} cluster center, μ_{ik} is the degree of membership of the k^{th} sample in the i^{th} cluster, and m is a constant greater than 1 (typically m = 2). The degree of membership, μ_{ik} is defined by

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{||x_k - v_i||}{||x_k - v_j||}\right)^{2/(m-1)}}$$

FCM will converge to a solution for v_i, that is either a local minimum or a saddle point of the cost function, J. The performance of the FCM solution depends strongly on the choice of the initial values used (eg. the number of clusters, c and the initial cluster centers, v_i), which are taken from the cluster estimation algorithm. Finally, the exact cluster membership can be computed by using the final iteration value of v_i.

VII. Performance Analysis

In this section, we evaluate the performance of the system and analyse its diagnostic capability.

A. Data Set

Collecting data from actual user complaints is challenging considering the amount of time required to collect a sufficient number of samples, and resources needed to manually identify the issues. In order to recreate a realistic fault detection scenario, we designed a fault emulator module to reproduce commonly found problems in UDs. The fault emulator is installed in test computers connected to the live university network in multiple locations. This allows us to demonstrate the viability of the system in real computing environments, where cross traffic and congestion is present. This method of emulating faults offered an efficient way of collecting accurate data with minimal resources.

A total of 16 common UD faults that can affect network performance are emulated as listed in Table II. By including the "Healthy" UD case, a total of 17 classes are formed and used in this evaluation. Over the entire evaluation period, we collected 12685 traces from UDs emulating these 17 fault cases, each having approximately

Fault	Description
CF1	Healthy
CF2	Disabled SACK error
CF3	Insufficient write buffer
CF4	Insufficient read buffer
CF5	Simultaneously insufficient read & write buffer
CF6	TCP timestamps are not working/in error
CF7	Window scaling error
CF8	Limited reordering threshold
CF9	Link-UD speed mismatch level 1
CF10	Link-UD speed mismatch level 1 & duplex mismatch
CF11	Link-UD speed mismatch level 2
CF12	Link-UD speed mismatch level 2 & duplex mismatch
CF13	UD firewall causing packet loss
CF14	UD firewall causing packet delay
CF15	Overloaded UD CPU
CF16	Overloaded UD memory
CF17	UD HDD i/o overloaded - faulty

TABLE II.
KEY: LIST OF FAULTS

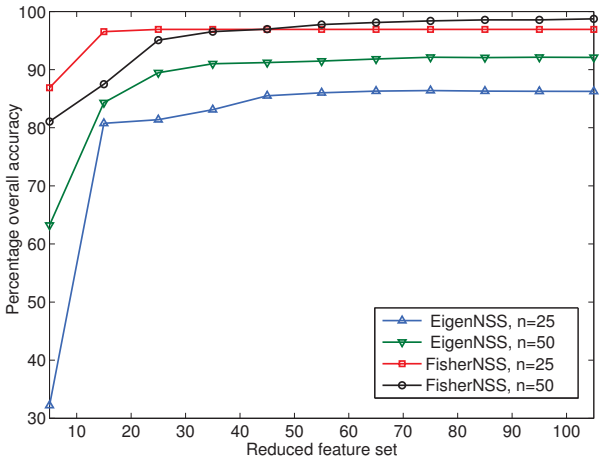


Figure 7. Overall accuracy of the system against dimensionality (D) of the reduced signatures. Each graph represents a different per-class training dataset size.

equal amounts (750) of samples. Data was also collected from UDs that operates with 8 different flavours of TCP, as they contribute to a variation in connection behaviour.

B. System Performance

The system is initially trained using only 13 classes as our evaluation needed to consider unknown faults. The faults CF2, CF8, CF10, and CF17 are kept as the unknown faults (refer to Table II), and are introduced at random intervals into the system during the testing stage. The data sets of the 13 classes are randomly divided into training and testing groups. We conducted the experiment over 8 iterative sessions and averaged the results in order to achieve statistical robustness.

The cluster threshold to add an unknown fault as a known fault are set to be equal to the number of per-class training samples used to initiate the system (e.g. if the system is initially trained with 50 samples per class, an unknown class is added as a known fault when its cluster membership reaches 50). This cluster membership minimum threshold is a design choice and will dictate the confidence level in detecting the existence of a new fault. Having a large cluster membership before categorizing them as a new fault improves the reliability of the system and yet, increases the time taken to offer users with a valid diagnosis.

Figure 7 shows the overall accuracy of the system in recognizing the testing samples which were previously unseen. Figure 8 shows the overall confusion rate of the system, which indicates the ratio between wrongly classified samples to the total samples. For both figures, n represents the number of samples used for each class during training, and the x-axis shows the dimensionality (D) of the EigenNSS and FisherNSS respectively. From Figure 7 and 8, we see that the system using FisherNSS managed to achieve a higher overall accuracy compared to the EigenNSS for any D transformed signature features and n-values. Any additional features used to construct the transformed signature only add a marginal performance gain. Noting that the original NSS contained 460 features, these results show a successful dimensionality reduction of 96.74%. Figure 7 shows that as the number of samples used for training increases, the system performance improves to a saturation limit. The overall accuracy gap between both EigenNSS-ED and FisherNSS-ED systems become closer as the number of training samples increases. Most importantly, the FisherNSS-ED system performs better than the EigenNSS-ED system when lesser numbers of training samples are used. This improvement is explained by adding to the system, and retraining with, correctly classified test samples. An overall accuracy of 80% and 20% confusion rate was achieved using EigenNSS with n=25 samples per class and D=15 reduced features for the 17 class system. Using FisherNSS, an overall accuracy of 97% and 3% confusion rate was achieved. FisherNSS is deemed to be the better system due to the fact that it requires lesser number of training samples to obtain a higher overall accuracy.

Table III summarizes the performance of the 17 classes used in our system. Metrics used in the table are as follows:

1) True-Positive Rate (TPR): Members of class X correctly classified as belonging to class X.
2) False-Positive Rate (FPR): Members of other classes incorrectly classified as belonging to class X.
3) True-Negative Rate (TNR): Members of class X incorrectly classified as belonging to other classes.
4) False-Negative Rate (FNR): Members of class X incorrectly classified as not belonging to class X.

Table III also shows that, all different types of faults can be uniquely identified independently with high-level of accuracy as suggested by high TPR and TNR. For example, faults CF6, CF7, CF12, CF13, CF14, and CF15 using both EigenNSS and FisherNSS have high TPR of about 90% and above. Most of the faults show a low FPR and FNR which indicates that the classifier has a low false detection rate.

Faults that were kept unknown to the system such as CF2, CF8, CF10, and CF17 only have a slightly smaller

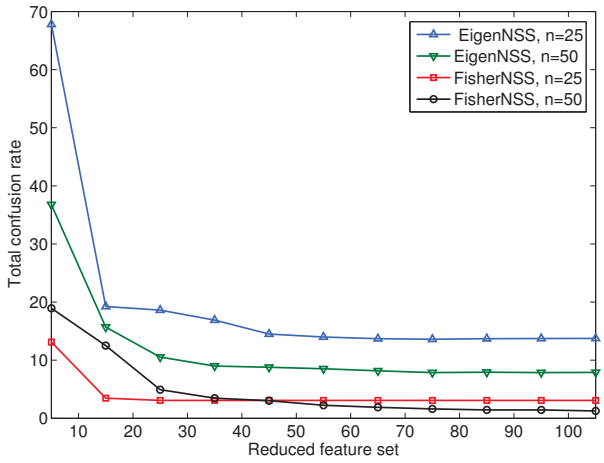


Figure 8. Overall confusion rate of the diagnostic system against dimensionality (D) of the reduced signatures for different per-class training dataset size (n).

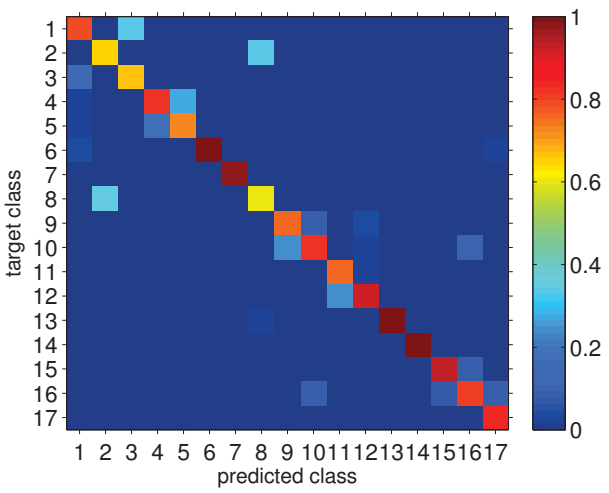


Figure 9. Confusion matrix for the 17 classes in EigenNSS-based diagnostic system.

TPR compared to other faults. This shows that the system has a high detection accuracy of unknown faults.

When EigenNSS are used, some of the faults such as CF1, CF2, CF3, CF5, CF8, CF9, and CF11 have relatively low TPR which makes them less likely to be correctly classified belonging to its respective class. However in some fault cases such as CF6, CF7, CF13, and CF15, EigenNSS has a better TPR than FisherNSS.

Figures 9 and 10 show the confusion matrix of the EigenNSS-based and FisherNSS-based system respectively. A confusion matrix is a typical form of visualization to observe the performance of an algorithm, in this case, the multiclass classifiers. The rows represent target or expected (actual) classes and the columns represent the predicted classes. The diagonal elements of the matrix represent the correct classifications whereas the other indices represent the incorrect instances. The ratio of each instance is color coded for better visualization. In Figure 10, the FisherNSS-based system manages to successfully avoid large misclassifications, satisfying a primary requirement of the system. However, Figure 9 shows a poorer performance due to the greater chance of misclassifications between classes for the EigenNSS-based system.

The proposed FisherNSS-based Euclidean distance classifier system (FisherNSS-ED) and the previously introduced systems (EigenNSS-ED and NSS-ED) are tested against a Naive Bayes multiclass classifier that used the original NSS data set (NSS-NB). The system is trained with all 17 classes at the beginning using similar training and testing sets of data. Figure 11 compares the overall system accuracy between the 4 systems, where the x-axis represents the number of training samples used. For any given number of training samples used, the figure shows that both EigenNSS-ED and FisherNSS-ED systems perform much better than the NSS-ED and NSS-NB systems. This is due to "over fitting" of classifiers used in the 460 feature NSSs, leading to degraded performance.

Fault	TPR		FPR		TNR		FNR	
	EigenNSS	FisherNSS	EigenNSS	FisherNSS	EigenNSS	FisherNSS	EigenNSS	FisherNSS
CF1	78.3	90.6	21.7	9.4	96.6	99.3	3.4	0.7
CF2	65.0	92.3	35.0	7.7	97.8	99.9	2.2	0.1
CF3	66.5	89.1	33.5	10.9	99.1	99.2	0.9	0.8
CF4	82.1	99.0	17.9	1.0	97.5	99.3	2.5	0.7
CF5	72.8	91.9	27.2	8.1	98.7	99.9	1.3	0.1
CF6	100	99.7	0	0.3	99.7	99.9	0.3	0.1
CF7	97.1	95.5	2.9	4.5	99.9	100.0	0.1	0.0
CF8	60.9	98.5	39.1	1.5	97.6	99.4	2.4	0.6
CF9	75.4	97.1	24.6	2.9	99.3	100.0	0.7	0.0
CF10	82.0	100	18.0	0	97.8	99.9	2.2	0.1
CF11	75.4	100	24.6	0	99.9	99.8	0.1	0.2
CF12	92.0	95.1	8.0	4.9	98.1	99.9	1.9	0.1
CF13	99.1	98.9	0.9	1.1	99.9	99.8	0.1	0.2
CF14	100	100	0	0	100	100.0	0	0.0
CF15	93.5	90.8	6.5	9.2	99.5	99.8	0.50	0.2
CF16	80.8	91.9	19.2	8.1	98.9	99.6	1.1	0.4
CF17	83.6	99.4	16.4	0.6	99.9	99.7	0.1	0.3

TABLE III.
PER-CLASS ESTIMATION PERFORMANCE OF THE EIGENNSS AND FISHERNSS BASED DIAGNOSTIC SYSTEMS AT $D = 15$ AND $n = 25$

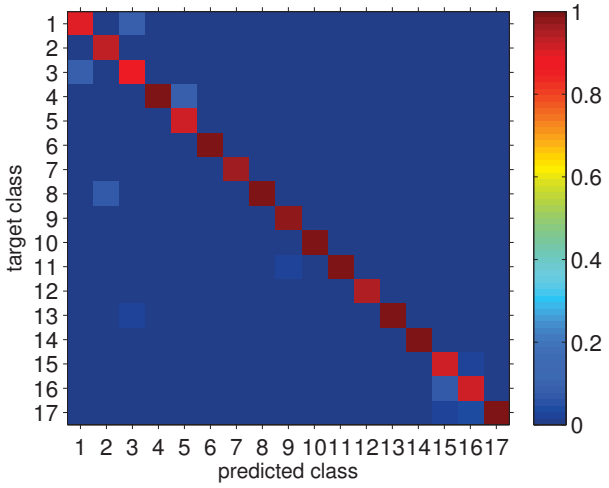


Figure 10. Confusion matrix for the 17 classes in FisherNSS-based diagnostic system.

Another important performance criteria for a system with iterative training process is the time taken to train the system and evaluate a new sample (diagnosis). We included this into the experiment to compare how both the total training time and a single diagnosis time vary with training samples per class for the previously mentioned classifiers. The NSS-NB classifier requires a much longer training time compared to the other classifiers as justified in Figure 12. The training time for the EigenNSS-ED classifier (24 ms-435 ms) is faster than the FisherNSS-ED classifier (33 ms-365 ms) when $n = 10 - 200$. Since both the training times are in the order of milliseconds, this suggests that iterative training does not impact the practical usability of either the EigenNSS-ED and FisherNSS-ED systems. The diagnosis time for a FisherNSS sample is in the order of microseconds (4 μs-59 μs), which is significantly faster than the other types of signatures, despite the fact that FisherNSS calculation involves more steps. This is followed by the EigenNSS-ED system which have a diagnosis time of about 4 μs-66 μs. This shows that both systems are not limited to an on-demand diagnosis, but could also be considered for "real-time" diagnosis applications. Real-time applications are often required to provide guaranteed response within a strict time constraint, usually in the order of milliseconds and sometimes even microseconds.

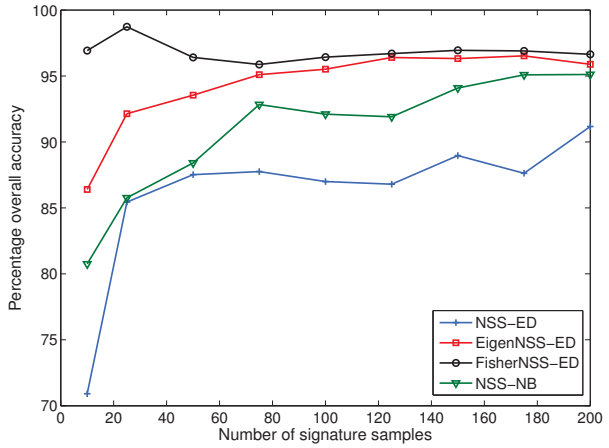


Figure 11. Comparison of overall accuracies of diagnostic systems.

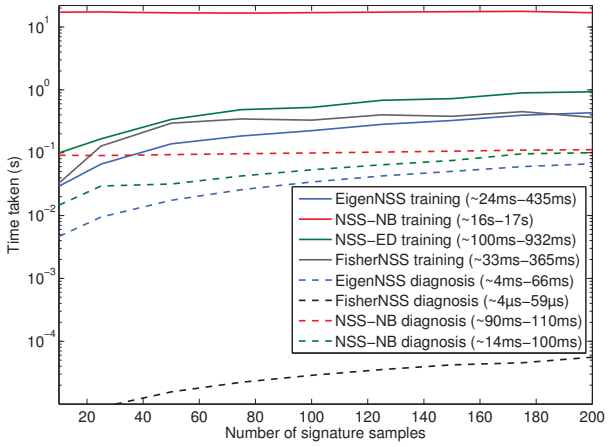


Figure 12. Training and single sample diagnosis time variations against increasing training dataset size for classifiers.

VIII. Conclusions

We have proposed and evaluated an automated UD soft-failure diagnostic system based on a single multi-class classifier design. The system is capable of diagnosing known and unknown faults by combining both supervised and unsupervised machine learning (ML) techniques. We have also presented another signature transformation technique to reduce the dimensionality of NSSs and also to remove network inconsistencies (unwanted information) from EigenNSS. This new transformed signature, FisherNSS aims to maximize the ratio of the between-class scatter matrix to the within-class scatter matrix to improve the classification process.

The system was evaluated by diagnosing 17 UD faults collected over a live campus network, achieving an overall accuracy of up to 97% using FisherNSS. When using FisherNSS, faults such as CF4, CF6, CF7, CF8, CF9, CF10, CF11, CF12, CF13, CF14, and CF17 have a high True Positive Rate of about 95% and above. We have also achieved a dimensionality reduction of 96.74% and low confusion rate between classes. Although the EigenNSS classifier have the shortest training time of about 24 ms-435 ms, the FisherNSS classifier is only marginally slower at about 33 ms-365 ms. Most importantly, FisherNSS samples have the shortest diagnosis time, in the order of microseconds of about 4 μs-66 μs compared to all the other types of samples.

This work provides the foundation to extend the system to a more sophisticated network environment with thousands of users, diverse client platforms and complex traffic patterns.

References

[1] S. Sundaresan, W. de Donato, and N. Feamster, "Broadband Internet Performance: A View From the Gateway,"

SIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 134–145, Aug. 2011.

[2] R. Maxion and F. Feather, "A Case Study of Ethernet Anomalies in a Distributed Computing Environment," *IEEE Trans. Rel.*, vol. 39, no. 4, pp. 433 –443, Oct. 1990.

[3] M. Mathis, J. Heffner, and R. Reddy, "Web100: Extended TCP Instrumentation for Research, Education and Diagnosis," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 69–79, 2003.

[4] S. Shalunov and R. Carlson, "Detecting Duplex Mismatch on Ethernet," in *Proceedings of PAM 05*. Boston, MA: Springer-Verlag, Berlin, Oct. 2005, pp. 135–148.

[5] C. Callegari, S. Giordano, M. Pagano, and T. Pepe, "Behavior Analysis of TCP Linux Variants," *Comput. Netw.*, vol. 56, no. 1, pp. 462–476, Jan. 2012.

[6] M. Thottan and C. Ji, "Anomaly Detection in IP Networks," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2191–2204, 2003.

[7] T. J. Hacker, B. D. Athey, and J. Sommerfield, "Experiences Using Web100 for End-to-end Network Performance Tuning," in *Proceedings of the 4th Visible Human Project Conference*, Nov. 2002.

[8] C. Widanapathirana, Y. A. Şekercioğlu, M. Ivanovich, P. Fitzpatrick, and J. Li, "Automated Inference System for End-To-End Diagnosis of Network Performance Issues in Client-Terminal Devices," *Int. Jour. of Comput. Netw. & Comm. (IJCNC)*, vol. 4, no. 3, pp. 37–56, 2012.

[9] C. Widanapathirana, J. Li, Y. A. Şekercioğlu, M. Ivanovich, and P. Fitzpatrick, "Intelligent Automated Diagnosis of Client Device Bottlenecks in Private Clouds," in *Proceedings of IEEE UCC 11*. Melbourne, Australia: IEEE, New York, Dec. 2011, pp. 261 –266.

[10] P. Sterlin, "Overfitting Prevention with Cross-Validation," Master's thesis, University Pierre and Marie Curie (Paris VI), Paris, France, 2007.

[11] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of intelligent and Fuzzy systems*, vol. 2, no. 3, pp. 267–278, 1994.

[12] J. C. Bezdek, "Fuzzy mathematics in pattern classification," *PhD Dissertion, Applied mathematics center, Cornell University*, 1973.

[13] C. Widanapathirana, J. Li, M. Ivanovich, P. Fitzpatrick, and A. Sekercioglu, "Automated diagnosis of known and unknown Soft-Failure in user devices using transformed signatures and single classifier architecture," in *38th Annual IEEE Conference on Local Computer Networks (LCN 2013)*, Sydney, Australia, Oct. 2013.

[14] C. Widanapathirana, J. C. Li, M. V. Ivanovich, P. G. Fitzpatrick, and Y. A. Şekercioğlu, "Adaptive Statistical Signatures of Network Soft-Failures in User Devices," *The Computer Journal*, p. bxt079, 2013.

[15] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, 1997.

[16] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification," in *Proceedings of SIGCOMM IMC 04*. Taormina, Italy: ACM, New York, Oct. 2004, pp. 135–148.

[17] B. Zhang, J. Yang, J. Wu, and Z. Wang, "MBST: Detecting Packet-Level Traffic Anomalies by Feature Stability," Comp. J., Advance Access, published January 5, 2012, Oxford, UK, 2012.

[18] T. Kihara, N. Tateishi, and S. Seto, "Evaluation of Network Fault-detection Method Based on Anomaly Detection With Matrix Eigenvector," in *Proceedings of APNOMS 11*. Taipei, Taiwan: IEEE, New York, Sep. 2011, pp. 1 –7.

[19] H. Hajji, "Statistical Analysis of Network Traffic for Adaptive Faults Detection," *Trans. Neur. Netw.*, vol. 16, no. 5, pp. 1053–1063, Sep. 2005. [Online]. Available: http://dx.doi.org/10.1109/TNN.2005.853414

[20] B. Aggarwal, R. Bhagwan, and T. Das, "NetPrints: Diagnosing Home Network Misconfigurations Using Shared Knowledge," in *Proceedings of USENIX NSDI 09*. Boston, Massachusetts: USENIX Association, CA, USA, Apr. 2009, pp. 349–364.

[21] T. Reidemeister, M. Jiang, and P. Ward, "Mining Unstructured Log Files for Recurrent Fault Diagnosis," in *Proceedings of IM 11*. Dublin, Ireland: IEEE/IFIP, New York, May 2011, pp. 377 –384.

[22] S. Lee and H. S. Kim, "End-user perspectives of internet connectivity problems," *Comput. Netw.*, vol. 56, no. 6, pp. 1710–1722, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2012.01.009

[23] H. Dahmouni, S. Vaton, and D. Rossé, "A Markovian Signature-Based Approach to IP Traffic Classification," in *Proceedings of MineNet 07*. San Diego, California, USA: ACM, New York, 2007, pp. 29–34.

[24] C. Manikopoulos and S. Papavassiliou, "Network Intrusion and Fault Detection: A Statistical Anomaly Approach," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 76 – 82, Oct. 2002.

[25] M. Wolfgang, *Host Discovery with nmap*, nmap.org, Palo Alto, CA, USA, Nov. 2002.

[26] J. V. Gomes, P. R. Incio, M. Pereira, M. M. Freire, and P. P. Monteiro, "Exploring Behavioral Patterns Through Entropy in Multimedia Peer-to-Peer Traffic," *Comp. J.*, vol. 55, no. 6, pp. 740–755, 2012.

[27] Z. Chen, Y. Zhang, Z. Chen, and A. Delis, "A Digest and Pattern Matching-Based Intrusion Detection Engine," *Comp. J.*, vol. 52, no. 6, pp. 699–723, Aug. 2009.

[28] R. Vaarandi, "A Data Clustering Algorithm for Mining Patterns from Event Logs," in *IP Operations and Management, 2003.(IPOM 2003). 3rd IEEE Workshop on*. IEEE, 2003, pp. 119–126.

[29] J. Yang and J.-y. Yang, "Why can lda be performed in pca transformed space?" *Pattern recognition*, vol. 36, no. 2, pp. 563–566, 2003.

[30] G. Donato, M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski, "Classifying facial actions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 10, pp. 974–989, 1999.

[31] C. Widanapathirana, J. Li, M. Ivanovich, P. Fitzpatrick, and Y. Şekercioğlu, "Adaptive Signatures of Soft-Failures in End-User Devices Using Aggregated TCP Statistics," in *Proceedings of IEEE/IFIP IM 13*. Ghent, Belgium: IEEE, New York, May 2013.

Chathuranga H. Widanapathirana is a Ph.D candidate in the Department of Electrical and Computer Systems Engineering at Monash University, Melbourne, Australia. He is also a data scientist working at Open Universities Australia specializing in analytics and big data. He received his B.Eng degree in Electronics with a major in Telecommunication Engineering at Multimedia University (MMU), Malaysia in 2009. His research interests include distributed cooperative networks, automated machine learning systems, data driven intelligent systems and end-user self-diagnostic services in multiuser networks.

X. Ang received the B.E degree in 2013 from Monash University and now studying towards a postgraduate degree in Electrical Engineering at the Department of Electrical and Computer Systems Engineering at Monash University, Melbourne, Australia

Jonathan C. Li received the B.E.in electrical and Electronic Engineering in 2001, B.Sc. degree in Computer Science and Information Technology Systems in 1999 from the University of Western Australia, and Ph.D. in Telecommunication from the University of Melbourne in 2010. He is currently a member of the academic staff at the Department of Electrical and Computer Systems Engineering of Monash University, Melbourne, Australia. His research interests are optical performance monitoring, routing in all-optical networks, network simulation and modeling, and Wireless TCP/IP optimization.

Milosh V. Ivanovich fills the role of Senior Emerging Technology Specialist within the Chief Technology Office of Telstra, and is an Honorary Research Fellow at Melbourne and Monash Universities in Australia. A Senior Member of IEEE, Milosh's interests lie in queuing theory, teletraffic modeling, performance analysis of wireless networks, and the study and enhancement of TCP/IP in hybrid fixed/wireless environments. Milosh obtained a B.E. (1st class Hons.) in Electrical and Computer Systems Engineering (1995), a Master of Computing (1996) and a Ph.D. in Information Technology (1998), all at Monash University.

Paul G. Fitzpatrick completed his Bachelor Degree in Electrical Engineering at Caulfield Institute of Technology, Melbourne in 1979 and his PhD in Electrical Engineering at Swinburne University, Melbourne in 1997 in the teletraffic performance of hierarchical wireless networks. Paul has over 30 years of experience working in the telecommunications industry and academia, including 15 years at Telstra Research Laboratories working on 2G, 3G and 4G wireless networks. His research interests focus on teletraffic modeling, quality of service, TCP performance modeling and analysis of telecommunication networks

Y. Ahmet Şekercioğlu is a member of the academic staff at the Department of Electrical and Computer Systems Engineering of Monash University, Melbourne, Australia. He has completed his Ph.D. degree at Swinburne University of Technology, and B.Sc., M.Sc. degrees at Middle East Technical University, Ankara, Turkey. He has lectured at Swinburne University of Technology, Melbourne, Australia for 8 years. His recent research interests are distributed algorithms for self-organization in wireless networks, application of intelligent techniques for multiservice networks as complex, distributed systems.