This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

netCSI: A Generic Fault Diagnosis Algorithm for Large-Scale Failures in Computer Networks

Srikar Tati *, Scott Rager *, Bong Jun Ko†, Guohong Cao*, Ananthram Swami‡, and Thomas La Porta *
*Pennsylvania State University, Email: {tati,str5004,gcao,tlp}@cse.psu.edu †IBM TJ Watson Research Centre, Email: bongjun_ko@us.ibm.com ‡Army Research Laboratory, Email: aswami@arl.army.mil

Abstract—We present a framework and a set of algorithms for determining faults in networks when large scale outages occur. The design principles of our algorithm, netCSI, are motivated by the fact that failures are geographically clustered in such cases. We address the challenge of determining faults with incomplete symptom information due to a limited number of reporting nodes. netCSI consists of two parts: a hypotheses generation algorithm, and a ranking algorithm. When constructing the hypothesis list of potential causes, we make novel use of positive and negative symptoms to improve the precision of the results. In addition, we propose *pruning* and *thresholding* along with a *dynamic threshold value selector*, to reduce the complexity of our algorithm. The ranking algorithm is based on conditional failure probability models that account for the geographic correlation of the network objects in clustered failures. We evaluate the performance of netCSI for networks with both random and realistic topologies. We compare the performance of netCSI with an existing fault diagnosis algorithm, MAX-COVERAGE, and demonstrate an average gain of 128% in accuracy for realistic topologies.

Keywords—fault diagnosis; large-scale network failures; incomplete information; clustered failures

1 Introduction

Efficiently detecting, diagnosing, and localizing faulty network elements [1], [2], [3] (e.g., network nodes, links) are critical steps in managing networks. The main focus of conventional fault detection and diagnosis techniques [1], [2] is to identify faults that are independent in nature, i.e., caused by individual component failures (e.g., routing software malfunction, broken links, adapter failure, etc.). In this paper, we concentrate on handling massive failures (or "outages") of many network elements caused by, for example, weapons of mass destruction (WMD) attacks [4], natural disasters [5], [6], electricity blackout, cyber attack, etc.

The type of outages caused by massive failures differ greatly from those caused by typical equipment faults [1], [2], [7]. Massive outages tend to create faults at multiple components that are geographically close to each other. We call these failures *clustered failures*. Until now, the prior work in the area of fault diagnosis has focused on independent failures [1], [3], [7]. The performance of these algorithms degrades when applied to clustered failures. In this paper, we propose netCSI, a new algorithm that is designed to effectively identify faulty network components under clustered failures. To show the benefits of our algorithm, we compare it with an existing algorithm that is proposed for independent failures.

netCSI determines possible causes of large-scale failures using a *knowledge base* and *end-to-end symptom* information. The knowledge base contains information about possible paths between different source-destination pairs and the inferred topology of the network. The end-to-end symptoms reflect end-to-end connectivity or disconnectivity in the network and are observed when a failure occurs. These symptoms include both *negative information*, such as which source-destination pairs are disconnected, as well as *positive information*, such as which source-destination pairs can still communicate.

Once a clustered failure occurs, netCSI uses the knowledge base and symptoms to generate a list of possible causes of the outage, called the *hypothesis list*. Then, a ranking algorithm is applied to the hypothesis list to rate the possible causes.

The main assumption in the existing fault diagnosis algorithms [1], [2], [7] is that complete and accurate information is available at the network manager. However, during large-scale failures, it is very unlikely that complete end-to-end symptom information will be available, because reporting nodes may not be able to reach the network manager. Hence, we specifically address the issue of incomplete end-to-end symptoms, because of a limited number of reporting nodes, or the inability of some of the reporting nodes to report all the symptoms. Note that netCSI considers the issue of incomplete symptoms and the aspect of positive information at the same time, unlike Steinder et al. [8], who consider these aspects separately.

The following are our key contributions in this paper:

- ***Utilizing positive information:*** Unlike existing fault diagnosis algorithms [1], [7], both *connectivity* and *disconnectivity* information are considered in netCSI instead of only *disconnectivity* information. This enables netCSI to generate a more precise hypothesis

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

list with fewer false positives. The average decrease in the false positives due to the addition of positive symptoms is 64% in our simulation study.

- ***Coping with lack of complete information:*** We understand the impact of partial symptom information on failure diagnosis. While generating the hypothesis list of possible causes, netCSI performs a type of exhaustive search, in which we introduce several optimization techniques that significantly decrease the run-time. Because of this, netCSI is able to generate much more accurate results than other well-known heuristic-based algorithms, such as MAX-COVERAGE [1], when there are few reporting nodes. In addition, we also look at the case when some of the reporting nodes do not have complete information. In our simulation performed on a realistic network topology, the average gain in the accuracy of netCSI over MAX-COVERAGE is 128% with limited reporting nodes.
- ***Conditional failure probability models:*** We develop conditional probability models based on characteristics of massive failures and the available topology of the network. These models are used in our ranking algorithm to enable accurate fault localization under clustered failures.

First, we review related work in Section 2. We describe our problem in Section 3, and explain netCSI and the ranking algorithm in Sections 4 and 5 respectively. We give an analysis of run-time in Section 6. We evaluate the performance of netCSI in Section 7. Finally, we provide conclusions in Section 8.

2 Related Work

There is much prior work [1], [3], [7] that address the localization of independent failures of individual components in the network. To the best of our knowledge, our algorithm is the first to focus on diagnosing massive failures and challenges posed by them. However, there is considerable interest on other aspects of large-scale failures in the current literature [9], [10].

Network tomography [11], [12] is a research area that has evolved over the past few years. The main focus of network tomography is inferring link-level properties from end-to-end measurements and network topology. One type of network tomography technique called Binary network tomography [13] deals with links that are only either good or bad, and is close to our problem. Most solutions in this area [14], [15] rely on large linear systems that represent the relation between path and link properties, which is *fundamentally underconstrained*, i.e., there exist links in the system whose properties or status cannot be determined uniquely. Some of the approaches [16], [17] try to solve this issue by employing statistical techniques that are based on maximum likelihood methods etc., but these still come short on accuracy when determining link states. On the contrary, netCSI does not solve a linear system of equations, but instead generates possible causes considering all links in the network.

In addition, network tomography assumes that there is a unique path between a source and destination pair in the network. As explained in [18], this could be a huge problem while diagnosing failures, when either rerouting occurs or there is possibility of dynamic topology in computer networks. Unlike tomography, netCSI considers all known multiple paths between a source and destination pair. Moreover, there is no work in network tomography that deals with large-scale failures, where a group of nodes fail together simultaneously, however there is a recent work that focuses on correlation in links while inferring the individual metrics [19].

MAX-COVERAGE [1], a fault diagnosis algorithm based on SCORE (space correlation engine) [7], focuses on MPLS-over-IP backbone networks. Localization agents use the end-to-end connectivity measurements as alarms and employ spatial correlation techniques to isolate failures. SCORE [7] is a greedy approach to localize optical link failures using only IP-layer event logs (IP link faults). The performance of these algorithms degrade for large scale failures under incomplete information. This problem of incomplete information is precisely addressed by netCSI. Furthermore, netCSI makes use of both negative and positive symptoms, unlike MAX-COVERAGE and SCORE.

Sherlock [3], Shrink [2], and Steinder et al [8] propose fault diagnosis techniques that use combinatorial algorithms with exponential computational complexity like netCSI. Sherlock and Shrink try to minimize the complexity by restricting the number of objects that are assumed to have failed, which introduces false negatives. However, they do not provide any mechanism to decide the number of objects that have to be restricted. We propose a dynamic threshold value selector (see Section 4.2.2), which uses a Bayesian approach to select the threshold. netCSI also uses a novel *pruning* technique to effectively reduce run-time without sacrificing accuracy. Furthermore, both Sherlock and Shrink assume complete symptoms, whereas netCSI performs well with incomplete symptom information.

3 Problem and Motivation

Our goal is to determine the list of faulty objects in a network as accurately as possible in the case of clustered failures, given a knowledge base comprising of network topology and end-to-end symptoms from a limited number of reporting nodes. We describe this system model and explore this problem further via an example in the following sub-sections.

3.1 System Model

We consider a network with n nodes, represented by the set $\mathcal{N}$, and l links, represented by the set $\mathcal{L}$. We define the term *objects* to represent either nodes or links. The set of $p = (n+l)$ objects in the network is denoted by O.

Each node communicates with destination nodes over different possible routes that are determined by routing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

algorithm [1]. These routes are loop-free and may not be equal to all possible routes in a given topology. *Source nodes* are represented by $S = \{s_1, s_2, s_3, ..., s_m\} \subseteq \mathcal{N}$. Different routes from a source node $s_i \in S$ to k different destinations, denoted by $D_i = \{d_1, d_2, ..., d_k\}$, are available in the knowledge base. The routes from source node s_i to a destination node $d_j \in D_i$ are given by $X_{i,j} = \{x^1_{i,j}, x^2_{i,j}, ...\}$. These routes are finite, and this number is different for each source-destination pair. The q-th route, $x^q_{i,j}$, is stored as the list of objects in O that are present along the route. If any of the objects along a route is faulty, then that route is disconnected; otherwise it is considered a connected route.

netCSI uses the end-to-end symptom information of the sources' connectivity and dis-connectivity to their respective destination sites. A destination site $d_j \in D_i$ is regarded as connected to s_i if there exists at least one connected route in $X_{i,j}$, and disconnected if all the routes in $X_{i,j}$ have at least one fault. This symptom information of a source s_i is given by the set $\mathcal{I}_i = \{\mathcal{I}_{i,1}, \mathcal{I}_{i,2}, \mathcal{I}_{i,3}, .., \mathcal{I}_{i,k}\}$, where the elements in the set correspond to the k destination nodes given in D_i. $\mathcal{I}_{i,j}$ is defined only for $d_j \in D_i$, and is equal to either 1 or 0 as follows,

$$\mathcal{I}_{i,j} = \begin{cases} 1 & \text{, if destination } d_j \text{ is connected to } s_i \\ 0 & \text{, if destination } d_j \text{ is disconnected to } s_i \end{cases}$$

The network manager collects these symptoms, I_i, from a reporting node $s_i \in S$. We assume that the collected symptoms are accurate, and have no errors. In large-scale failures, getting symptom information from every reporting node in the network is a major challenge. In addition, the collection of symptom information involves high overhead in terms of network management traffic. Note that netCSI could diagnose both node and link failures depending on the objects that have been specified in routes by the network manager. In the next section, we consider only nodes in the routes, however in our simulations (Section 7.1), we consider links in the network as the objects in routes.

3.2 Motivating example

(SRC,DEST)	Routes
(s_1, d_1)	$r^1_{s_1,d_1} : (R_1, R_2)$ $r^2_{s_1,d_1} : (R_1, R_3)$
(s_1, d_2)	$r^1_{s_1,d_2} : (R_1, R_3)$
(s_2, d_1)	$r^1_{s_2,d_1} : (R_4, R_3)$
(s_2, d_2)	$r^1_{s_2,d_2} : (R_4, R_3)$ $r^2_{s_2,d_2} : (R_4, R_5)$

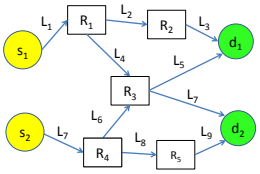


Fig. 1: A network scenario to illustrate the algorithm

Consider the network in Figure 1, in which there are two sources, s_1 and s_2, each connected to two destination nodes, d_1 and d_2. The possible routes for all source-destination pairs are given in a table in Figure 1. In this example, we only consider the intermediate nodes as objects in the network. Suppose we are given the following symptoms: source node s_1 is disconnected from both the destination nodes d_1 and d_2, and source node s_2 is disconnected from destination node d_1, but is connected to destination node d_2.

First consider the case in which we look only at the set of negative symptoms, $\{I_{1,1} = 0, I_{1,2} = 0, I_{2,1} = 0\}$. In this case, there are eight possible causes of symptoms as shown in Table 1. Up to this point, we cannot say that any particular router is down with complete confidence because of the ambiguity in the possible causes deduced from the given negative symptoms. However, if we know that routers R_1 and R_3 are geographically close to each other, for instance in the same building, we can consider the double failure (R_1, R_3) as the most likely possible cause of the observed symptoms.

No. of failed objects	Negative Information from s_1	Positive and Negative Information from s_1
1-object	(R_1)	(R_1)
2-objects	$(R_1, R_2), (R_2, R_3), (R_1, R_3)$	$(R_1, R_2), (R_2, R_3), (R_1, R_3)$
3-objects	(R_1, R_2, R_3)	(R_1, R_2, R_3)
No. of failed objects	Negative Information from s_2	Positive and Negative Information from s_2
1-object	$(R_4), (R_3)$	(R_3)
2-objects	(R_4, R_3)	()
No. of failed objects	Negative Information from s_1 and s_2	Positive and Negative Information from s_1 and s_2
1-object	()	()
2-objects	$(R_1, R_3), (R_1, R_4), (R_2, R_3)$	$(R_1, R_3), (R_2, R_3)$
3-objects	$(R_1, R_2, R_3), (R_1, R_2, R_4)$ $(R_1, R_3, R_4), (R_2, R_3, R_4)$	(R_1, R_2, R_3)
4-objects	(R_1, R_2, R_3, R_4)	()

TABLE 1: Possible causes of given symptoms

If, in addition to negative information, we also have the positive information from source nodes s_1 and s_2 (in this case $I_{2,2} = 1$), we can infer three possible causes as shown in Table 1. From these causes, we can deduce that router R_3 must be down and router R_4 must be operating correctly. If we again know that routers R_1 and R_3 are close to each other, then we can prioritize the double failure (R_1, R_3) as the most likely possible cause. As seen in this example, the addition of positive information decreases the number of possible causes. Moreover, with the knowledge of geographical topology information, we can rank the possible causes of symptoms to enable more accurate localization.

In addition, if we compare the cases where positive and negative symptoms is acquired from only s_1 (incomplete symptoms) versus both the source nodes s_1 and s_2 (complete symptoms) in table 1, the latter case results in fewer possible causes. In other words, having symptoms from more reporting nodes generally results in fewer false positives and higher accuracy. However, the reduction in false positives comes with added overhead in the form of extra network management traffic. Therefore, this trade-off is important when there are limited reporting nodes in the network and is discussed in more detail in Sections 7.4.3.

1. Our fault diagnosis algorithm is agnostic of the routing algorithm which is chosen by the network manager

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

4 NETCSI:HYPOTHESES GENERATION ALGORITHM AND REFINEMENTS

Given the knowledge base and end-to-end symptoms of the network as input, we propose a hypotheses generation algorithm that outputs possible combinations of faulty objects that can cause the observed failures. We then introduce several heuristics to decrease the run-time of the hypotheses generation algorithm.

4.1 Hypotheses Generation Algorithm

Step I: A list of *possible faulty objects* is constructed in this step. The list is comprised of all unique objects in routes from different sources (reporting nodes) to the corresponding disconnected nodes. We define a set with all possible faulty objects, represented by $\mathcal{P}$. Consider all source-destination pairs with negative information, i.e., $I_{s_i,d_j} = 0$ for a (s_i, d_j) pair, with $d_j \in D_i$. The unique objects in all possible routes between different source and destination pairs are added to the faulty object list. The sample faulty object list for the example shown in Figure 1 and the symptoms mentioned in Section 3.2 is (R_1, R_2, R_3, R_4). Step I is implemented in lines 2-11 of the pseudo-code shown in Figure 2.

Step II: Every combination of possible faulty objects in the formulated list is checked for the given symptoms utilizing the knowledge base. We eliminate the combinations of faulty objects which make observed symptoms impossible and shortlist the rest of the combinations of faulty objects as potential causes of symptoms. This shortlist of possible combinations is called the *hypothesis list*.

We consider two kinds of symptom information in this algorithm: negative and positive. Note that this step can be parallelized, i.e., different combinations can be processed in parallel to check their feasibility for the given end-to-end symptoms. Lines 24-45 show the implementation of step II in the pseudo-code given in Figure 2.

If there are N objects in the possible faulty object list, then the search space is $2^N - 1$. Thus, the computational complexity grows exponentially in N. To decrease the run-time of the algorithm, we present refinements to the base netCSI algorithm that utilize both positive and negative symptoms below.

4.2 Decreasing Computational Complexity

4.2.1 Pruning

We can decrease the run-time of netCSI by reducing the search space of combinations. The underlying idea of *pruning* is to recognize the combinations that are not possible because they disconnect all the routes in $X_{i,j}$ between source $s_i \in S$ and destination $d_j \in D_i$, which are actually connected according to the given positive symptoms, i.e., $I_{i,j} = 1$. We include all such combinations in the *Impossible Set*, represented by IS.

Since the combinations of faulty objects in IS are not possible causes of failure, any other combination that includes at least the same faulty objects as a combination in IS cannot be a possible cause of failure. Therefore, Step II of the hypotheses generation algorithm, can be avoided in the case of any combination which is a superset of a combination in IS.

Note that the advantage of this pruning can be realized completely only when combinations are searched in ascending order of cardinality, i.e., the combinations with one faulty object are considered first, followed by combinations with increasing numbers of faulty objects, since combinations with the lowest cardinality have the highest number of supersets. Pruning can be applied to the base netCSI algorithm. We call this approach $OPT1$. In our simulations, the average reduction in the search space of $OPT1$ is 94% when compared to base netCSI.

4.2.2 Thresholding

The motivation for *thresholding* also comes from the objective to decrease the search space of the combinations of possible faulty objects. In thresholding, we eliminate some combinations of objects from the search space with the help of certain pre-defined attributes, depending on the circumstances and nature of the network [2] [3].

Attributes can be the number of faulty objects in the combination [3], threshold values for the joint probability of the combination, etc. For example, if the attribute is the number of faulty objects, then the combinations where the number of faulty objects is higher than the threshold value are eliminated from the search space to decrease the number of computations required.

Both pruning and thresholding can be simultaneously applied to the base netCSI algorithm. We call this approach $OPT2$. In our experiments, we found that $OPT2$ with a proper threshold value that does not cause any loss in accuracy can reduce the search space by 97.5% when compared to base netCSI. Note that the netCSI has exponential computational complexity as we explained before, therefore an extra reduction of 3.5% with thresholding is very significant. We will assess this trade-off in our experimental results in Section 7.4.1.

Unlike pruning, thresholding reduces the search space with a potential loss in accuracy of diagnosis: the hypothesis list might lose some combinations, i.e., it might be less accurate. However, with an appropriate threshold value, thresholding does not change the accuracy.

To appropriately set this value, we devise a new technique called dynamic threshold value selection. We find that the appropriate value is close to the expected number of faults that can occur due to a large scale failure, i.e., we have to choose a different threshold value in different failure scenarios. We estimate the number of failures based on symptoms in the network to select an appropriate threshold value.

Dynamic threshold value selection: This technique is based on Bayesian decision approach. We make use of both training data and the current observed negative symptoms when a failure occurs, to estimate the expected number of failures. We then use this number as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

the threshold value.

We consider training data with different samples of failures, and stores the resultant number of negative symptoms for each sample along with its frequency of occurrence, i.e., a histogram with number of negative symptoms as an independent variable, which is represented by A. This histogram data is divided into different bins that represent different ranges of failures, i.e., in terms of number of failures. The total number of bins is represented by b, and each bin is indexed by i.

We estimate the probability density function, CF_i, for each bin i using kernel density estimation [20]. Thus, the training data will result in different probability density functions corresponding to various bins. In addition, the training data consists of both offline and online failure data. The offline failure data is acquired by choosing random samples of clusters in the network and then using network measures as explained in the next paragraph. The online failure data is recorded during the occurrence of actual failures.

For offline failure data, we consider both independent and clustered failures. We use the network measures betweenness centrality (BC) and group betweenness centrality (GBC), to determine the end-to-end negative symptoms that will result from a failure instead of enumerating the symptoms by simulating all possible failures in the network. BC for an object is defined as the fraction of paths that pass through the object, out of all the paths between different nodes in the network. Conventionally, BC is based on single shortest paths in a given network; here the definition is with respect to all pre-failure paths, $X_{ij}, i \in S, j \in D_i$ that are available to the network manager. GBC of a cluster is defined as the fraction of paths that pass through that particular cluster out of all the paths between different nodes in the network. In the above definitions, the considered paths between nodes are the ones which are available to the network manager from reporting nodes. Note that the paths are considered to be loop-free here.

Since we consider multiple paths between source and destination nodes, in a single object failure case, the number of negative symptoms is approximately equal to BC of the failed object. Moreover for independent failures, when multiple objects fail simultaneously, the total number of end-to-end negative symptoms is estimated as the sum of the BC values of the failed objects.

For clustered failures, we choose samples that include both random cluster and random space failures (Section 7.1) depending on the available cluster information at the network manager. For both modes, the number of end-to-end negative symptoms is equal to cluster's GBC value.

When a large-scale failure occurs in the network, we observe the number of negative symptoms at a failure instance, ns. Given ns, and the probability density functions of various bins (the histogram discussed earlier in this section), we use a Bayesian approach to identify the size of the failure set. We then use this as the threshold value. Our technique is described in Equations 1 and 2. The prior probability, $P(CF_i)$, for a bin i is evaluated using training data. In the absence of training data, uniform priors are assumed.

$$P(CF_i|A=ns) = \frac{P(A=ns|CF_i)P(CF_i)}{\sum_{i=1}^{b} P(A=ns|CF_i)P(CF_i)} \quad (1)$$

$$\text{Selected Bin Number} = \underset{i}{\text{argmax}} P(CF_i|A=ns) \quad (2)$$

We evaluate netCSI with dynamic threshold value selection in Section 7.4.6.

5 NETCSI: RANKING

In this section, we present a *ranking algorithm* that utilizes a *conditional failure probability model* to rank the hypothesis list.

5.1 Ranking Algorithm

Once the hypothesis list is available, we employ a ranking algorithm to rate the possible combinations based on the *maximum likelihood principle*. The combinations in the hypothesis list are ranked in the decreasing order of their joint probabilities. Here, the joint probability of a combination is the probability that the events corresponding to both faulty and non-faulty objects will happen at the same time.

To demonstrate the idea behind the computation of joint probability, we look at a general combination with both faulty and non-faulty objects, i.e., a combination of N objects in $\mathcal{P}$, a set of possible faulty objects (Step I in Section 4.1) that has arbitrary sets of k faulty and $N-k$ non-faulty objects. We denote the set of faulty objects as $\mathcal{F} = \{o_1, o_2, ..., o_k\}$ and the set of non-faulty objects as $\mathcal{S} = \{o_{k+1}, o_{k+2}, ..., o_N\}$. Therefore by construction: $\mathcal{F}, \mathcal{S} \subset \mathcal{P}$, $\mathcal{F} \cup \mathcal{S} = \mathcal{P}$, and $\mathcal{F} \cap \mathcal{S} = \emptyset$. The failure event of an object $o_i \in \mathcal{P}$ is modeled using a random variable, F_i. The value of F_i is either 1 if o_i is faulty, or 0 if o_i is not faulty. The expression for the joint probability of objects $(o_1, o_2, .., o_k)$ being faulty and objects $(o_{k+1}, o_{k+2}, .., o_N)$ being not faulty is,

$$\begin{aligned} P(F_1 = 1, .., F_k = 1, F_{k+1} = 0, .., F_N = 0) = \\ P(F_1 = 1) * P(F_2 = 1|F_1 = 1) * \\ .. * P(F_k = 1|F_1 = 1, F_2 = 1, .., F_{k-1} = 1) * \\ P(F_{k+1} = 0|F_1 = 1, F_2 = 1, .., F_k = 1) * .. * \\ P(F_N = 0|F_1 = 1, .., F_k = 1, .., F_{N-1} = 0) \end{aligned} \quad (3)$$

The individual conditional probabilities on the R.H.S. of equation 3 are computed with the help of the conditional failure probability model as explained in the next section.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing


```
{STEP-I}
for i = 1 to numReportingNodes do
    for j = 1 to numDestinations(i) do
        if I_{i,j} == 0 then
            add all objects of x^1_{i,j}, .., x_{i,j}^m to
                possibleFaultyObjectsList
            add (i, j) to negativeSymptoms
        else
            add (i, j) to positiveSymptoms
        end if
    end for
end for
{Combs is combinations}
{numPossibleFaultyObjects is N}
th=N (with no opt-2)
{opt-2} & th=thresholdValue
for k = 1 to th do
    numObjectCombs(k)=^N C_th
    {opt-1}
    if any ObjectCombs(k)⊇impossibleSetCombs(IS)
    then
        remove from ObjectCombs(k)
    end if
    numFaultyObjectCombs=numFaultyObjectCombs
        + numObjectCombs(k)
    possibleComb(AllObjectCombs(k))=1
    {STEP-II–PARALLELIZATION}
    for i = 1 to numObjectCombs(k) do
        {Positive Symptom Information}
        for j = 1 to numPositiveSymptom (c) do
            if positiveSymptom(j) does NOT hold for
            faultyObjectComb(i) then
                possibleComb(i)= 0
                add faultyObjectComb(i) to
                    impossibleSetCombs(IS)
                BREAK
            end if
        end for
        {Negative Symptom Information}
        for j = 1 to numNegativeSymptom (d) do
            if negativeSymptom(j) does NOT hold for
            faultyObjectComb(i) then
                possibleComb(i)= 0
                BREAK
            end if
        end for
        if possibleComb(i)== 1 then
            {Calculate Joint Probability for the Ranking
            Algorithm}
            add presentObjectCombs(i) to
                possibleFaultyCombs (e)
        end if
    end for
end for
```

Fig. 2: Pseudo-code

5.2 Conditional Failure Probability Models

The main objective of Conditional Failure Probability (CFP) models is to determine the values of conditional probability that an object in an arbitrary set is faulty or not given the faultiness of other objects in the same set. These values are used to compute the joint probability of a combination as shown in equation 3.

Here, we consider that the network may be divided into different clusters based on various attributes of the network. For example, objects in the same building may be considered to be in one cluster or objects enclosed in a specified geographical area may be labeled as one cluster. Given clustered failures, it is most likely that all objects in a cluster may be either faulty or not faulty collectively.

Alternatively, we can also consider that objects within a certain distance metric from each other, e.g., physical distance or hop count, are most likely to fail together. In this context, there can be three possible kinds of information: cluster information (CI), object distance information (OD), and no information (NI). We do not describe the conditional failure probability model with no information (CFPM-NI) here because it is straightforward, and details can be found in [21].

5.2.1 Conditional failure probability model with cluster information (CFPM-CI)

In CFPM-CI, we consider that there are c clusters in total; the number of objects in each cluster can vary. We represent the cluster of object o_i as C_i. We consider two arbitrary sets of faulty and non-faulty objects, represented by $\mathcal{F}'$ and $\mathcal{S}'$, respectively. As before, $\mathcal{F}', \mathcal{S}' \subset \mathcal{P}$ and $\mathcal{F}' \cap \mathcal{S}' = \emptyset$. The probability of an object o_i being faulty conditioned over other remaining faulty and non-faulty objects is given in equation 4 and the conditional probability that an object o_i is not faulty given $\mathcal{F}'$ and $\mathcal{S}'$, $P(F_i = 0|\mathcal{F}', \mathcal{S}')$, is simply the complement of the probability given in equation 4.

$$P(F_i = 1|\mathcal{F}', \mathcal{S}') = \begin{cases} p_c, \text{if } \exists\ o_j \in \mathcal{F}' : C_i = C_j \wedge j \neq i \\ 1 - p_c, \text{otherwise} \end{cases} \tag{4}$$

The above equation signifies that if any of the objects in the same cluster as object o_i is faulty then the object o_i is faulty with a probability p_c. Since we are dealing with clustered failure scenarios, p_c should be very close to 1.

We consider homogeneity in clusters when it comes vulnerability to clustered failures, and hence we propose a constant value for p_c. If we know information about the vulnerability of all clusters for clustered failures, then we can introduce variable values for p_c and extend our model to that setup.

5.2.2 Conditional failure probability model with object distance information (CFPM-OD)

Here, we deal with a situation where no information about clusters is known, but some metric of distance between objects in the network is known. The distance between objects can be measured through various parameters, for example euclidean distance, hop distance, etc. We define the distance between two objects o_i and o_j as a function $d(i, j)$.

As explained above, we consider two arbitrary sets of faulty and non-faulty objects, $\mathcal{F}'$ and $\mathcal{S}'$. The conditional probability that an object o_i is faulty given the remaining faulty and non-faulty objects is defined in equation 5.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

Again, the conditional probability that an object o_i is not faulty, $P(F_i = 0|\mathcal{F}', \mathcal{S}')$, is just the complement of the probability given in equation 5. Here, the function $d(i, \mathcal{F}')$ in equation 5 is defined as the minimum distance between the object o_i and any object of the set $\mathcal{F}'$ because the event that an object is faulty or not faulty depends on the closest failed object.

$$P(F_i = 1|\mathcal{F}', \mathcal{S}') = p_o^{d(i,\mathcal{F}')} \quad (5)$$

where $p_o \in [0, 1]$, and $d(i, \mathcal{F}') = min(d(i, j))\ \forall o_j \in \mathcal{F}'$ and $o_j \neq o_i$. In this paper, hop distance between objects is used as the distance measure in the simulations.

6 Run-time Analysis

For this analysis, we define the term *Search Space* to represent the number of combinations for which the hypotheses generation and ranking algorithms of netCSI must be applied; $SS_{POS+NEG}$ represents the search space of base netCSI and SS_{OPT1} represents the search space for the refinement $OPT1$. Also, let m be the minimum cardinality of all the combinations in the Impossible Set (IS) for a given scenario. $SS_{POS+NEG}$ is equal to $2^N - 1$, where N is the number of possible faulty objects. N depends on the number of objects (nodes or links) in the network and number of routes considered by netCSI. Therefore, this analysis offers an understanding on how netCSI scales with larger networks.

With pruning, portions of the search space will be eliminated for any scenario with a non-empty IS. Specifically, we can define upper and lower bounds on SS_{OPT1} for any given m and non-empty IS. The upper bound is achieved when IS has only one combination with m objects, and the lower bound occurs when IS has all possible combinations with m objects. These upper and lower bounds of SS_{OPT1} are shown in equation 6,

$$\sum_{i=0}^{m} \binom{N}{i} - 1 \leq SS_{OPT1} \leq (2^N - 1) - (2^{N-m} - 1) \quad (6)$$

In the case of thresholding, with an arbitrary threshold value, th, the original search space of $2^N - 1$ is reduced to $\Gamma = (2^N - 1) - (\binom{N}{th+1} + \binom{N}{th+2} + ... + \binom{N}{N})$. Since $OPT2$ utilizes both pruning and thresholding, we can follow a logic similar to that is used for $OPT1$ to evaluate the bounds of SS_{OPT2}, using Γ as the search space instead of $2^N - 1$.

In order to see the benefit of pruning more easily, we can look at the fractional reduced search space, which lies in the interval $[\frac{\sum_{i=0}^{m}\binom{N}{i} - 1}{2^N - 1}, \frac{(2^N - 1) - (2^{N-m} - 1)}{2^N - 1}]$. For example, with $m = 1$ and $N = 10$, the search space is reduced to as low as 0.98% and as high as 50% of its original size. From our experiments in Section 7.4, we picked 10 cases with different numbers of possible faulty objects (N). In all cases, the minimum cardinality of all the combinations in IS is 1, i.e., $m = 1$. From Figure 3, we see that the experimental values are close to the upper limits of the reduction in search space, resulting in a *search space of less than 1%* of that which is processed by the base netCSI algorithm.

Please note that the lower bound of SS_{OPT1} can be reduced to a polynomial (N^m) for small values of m, and it is a best-case scenario when all possible combinations with m objects are present in IS. This can be a useful insight when the scale of the network is increased, i.e., increasing the value of N. However, the determination of value m for large networks is non-trivial and depends on scale of failures primarily. In addition, the value m depends on the number and the kind of positive and negative symptoms that are available in the network.

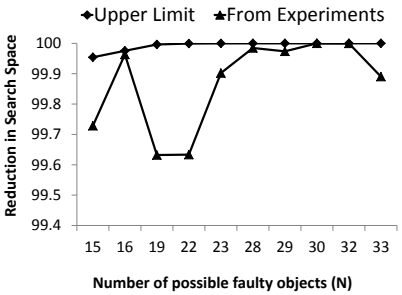


Fig. 3: Reduction in search space of $OPT1$ for different N with $m = 1$

7 Evaluation And Results

7.1 Experimental Setup

We evaluate the performance of netCSI on two different types of topologies, random and realistic. The random networks are generated by placing nodes at random locations in a 2-dimensional space and the links are generated between the nodes like a waxman random graph [22]. The realistic topology is taken from Rocketfuel trace data provided by the University of Washington [23]. A network with 315 backbone routers that are connected by 972 links in an ISP (Sprint in US).

In both the networks, multiple paths are generated among all nodes using Dijkstra's algorithm. We consider a maximum of 5 paths between each pair of nodes in the network. These paths are generated after arbitrarily failing certain links in the network, and are of varying lengths. Without loss of generality, we only consider links as objects in a path. In every experiment, a certain number of reporting nodes is selected randomly out of all nodes to report end-to-end symptoms.

For simulation purposes, clustering is done in different ways for random networks and the network with realistic topology. In random networks, the network is divided into different disjoint clusters using the k-means clustering algorithm [24]. With the number of clusters (k) as the input, the k-means clustering algorithm generates k clusters, each with variable number of nodes. For the network with realistic topology, all the nodes in a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

particular city are regarded as one cluster; there are 24 clusters in total.

We simulate two types of failure modes: *random cluster failure* and *random space failure*. In the random cluster failure mode, we choose a random cluster out of all clusters in the network and fail all objects in that particular cluster. This failure mode mimics a realistic scenario where all objects in a building (considered as one cluster) may fail due to various reasons. For random space failure mode, a geometric circle with a prescribed radius r is randomly chosen, and all objects within that circle are failed. This scenario is similar to a case where objects that are geographically adjacent in multiple clusters can fail together, or a group of objects which are geographically close in a large wireless network, can fail simultaneously. Note that in Rocketfuel topologies, we simulate only random cluster failures and not random space failures.

We parallelized the code of netCSI, and used an 8 core processor to improve the run-time. Since the fault diagnosis is done at the network manager in our case, we expect that netCSI can run on an advanced infrastructure in practice.

7.2 Performance Metrics

7.2.1 Hypotheses Generation Algorithm

Several metrics can be used to measure the performance of the hypotheses generation algorithm. The two metrics are *precision* and *accuracy*. Precision of the hypothesis list is defined as the size of the hypothesis list. A smaller hypothesis list is said to be more precise, since the network manager has to deal with fewer possible causes of failure based on symptoms. Accuracy of the hypothesis list is defined as the fraction of faulty objects that are present in the ground truth. More details are given in Section 7.2.3. In addition, we also consider the *size of the search space*, which is directly related to the time complexity of the hypotheses generation algorithm, and thus determines the run-time of netCSI.

7.2.2 Ranking Algorithm

In the context of inputs given to netCSI, we define the ground truth as the set of all objects that have failed which are also present in at least one path from any given reporting node. The set of faulty objects in the ground truth is defined as *Actual Faulty Objects* (AFO).

We define *cumulative rank* (CR) to evaluate the performance of the ranking algorithm. We consider only a subset of objects as faulty objects instead of considering all objects by using the ranks of different combinations in the hypotheses list. In that context, CR is a particular rank above which the objects in the *union* of combinations at higher ranks (lower numeric values of rank) are treated as faulty objects. *Cumulative rank with ground truth* (CR_{GT}) is a special value of CR that is defined as the lowest rank above which all faulty objects of the ground truth are accumulated.

For example, consider a sample hypothesis list with 7 combinations that is ranked as shown in Table 2. Let the ground truth be the combination (o_1, o_2, o_4, o_5). In this example, the exact rank of the ground truth is 5, whereas CR_{GT} is 3. The faulty object set above CR_{GT} is $O_1 \cup O_2 \cup O_3 = (o_1, o_2, o_3, o_4, o_5)$. This set is called *Cumulative Faulty Objects* ($CFO_{CR_{GT}}$), and can be defined for any cumulative rank (CFO_{CR}).

hypothesis list			
Rank	Combination of objects	Rank	Combination of objects
1.	$O_1 = (o_1, o_2)$	**5.**	$\boldsymbol{O_5 = (o_1, o_2, o_4, o_5)}$
2.	$O_2 = (o_1, o_3)$	6.	$O_6 = (o_1, o_2, o_3, o_6)$
3.	$O_3 = (o_4, o_5)$	7.	$O_7 = (o_1, o_2, o_3, o_4, o_6)$
4.	$O_4 = (o_1, o_3, o_5)$		

TABLE 2: A sample hypothesis list

7.2.3 False Positives and False Negatives

False Positives and False Negatives are possible with respect to both the hypothesis list and CR. False Positives with respect to CR (FP_{CR}) are objects which are present in CFO_{CR}, but not in AFO. False Negatives with respect to CR (FN_{CR}) are objects which are part of AFO, but not in CFO_{CR}. The expressions for $\%FP_{CR}$ and $\%FN_{CR}$ are given below. In addition, we also define accuracy for fault diagnosis algorithms as the fraction of objects in the ground truth that are diagnosed as faulty, i.e., accuracy is the *complement* of FN_{CR}. Different CR values give distinct CFO_{CR}, and thus different false positives and false negatives. The effect of CR over accuracy, false positives, and false negatives will be seen in more detail in Section 7.4.

$$\%FP_{CR} = \frac{|CFO_{CR} - AFO| * 100}{|CFO_{CR}|}$$

$$\%FN_{CR} = \frac{|AFO - CFO_{CR}| * 100}{|AFO|}$$

7.3 Robustness of CFP Models

To understand the robustness of CFP models, we have run simulations on a random network with 100 nodes, $k = 20$ clusters and 10 randomly selected reporting nodes. $OPT1$ is employed for fault diagnosis. All three CFP models are compared under two failure modes, random space and random cluster failures. We evaluate the metric average exact match rank, which is defined as the rank of the ground truth in the ordered hypothesis list. The effect of parameters p_c and p_o (equations 4 and 5) on the ranking algorithm is examined by choosing values from the set $\{0.5,0.9\}$. We consider constant values for these parameters, but p_c and p_o could be adaptively learned from occurrences of clustered failures in the network. This is considered as a future work.

Figure 4 shows that the performance of the ranking algorithm using the CFPM-CI dominates both CFPM-OD and model with no information. The CFPM-CI performs better when $p_c = 0.9$ in both the failure modes. However, CFPM-OD performs better when $p_o = 0.5$ compared to 0.9 in the case of random space failure mode where $r^2 = 20$. This is not true in the case where $r^2 = 40$. The reason

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

is that the case $r^2 = 20$ represents a smaller clustered failure scenario compared to the scenario where $r^2 = 40$. Here, r is the radius of the circle. A similar observation can be made for random cluster failure scenarios, when $k = 30$ (smaller cluster size) compared to the case when $k = 10$ (bigger cluster size) as shown in Figure 4. Here k is the number of clusters. Regardless of the failure modes, CFPM-CI with $p_c = 0.9$ is the most robust of all the models. Thus, we use this model for all subsequent simulations in Section 7.4. Nevertheless, having object distance information is still better than not having any information.

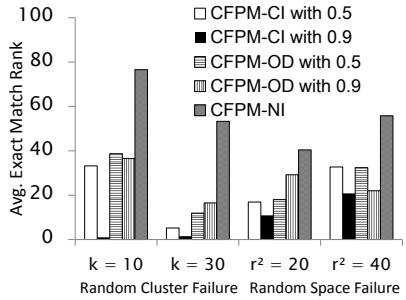


Fig. 4: Robustness of CFP models

7.4 Evaluation Results

In this section, we present the results to gauge the performance of netCSI, and also compare it with MAX-COVERAGE. The experiments are run by assuming that the network manager has complete knowledge of the multiple paths that are consistent with the assumptions given in Section 3.1, and the network topology. This is a standard assumption in prior works such as [1], [2], [3], [7] including MAX-COVERAGE and is a valid assumption if the network manager owns the network.

7.4.1 Performance of different netCSI approaches

Here, we analyze the performance of netCSI with both positive and negative information ($POS + NEG$), and only negative information (NEG). For the former case, we evaluate the improvements that can be achieved with both $OPT1$ and $OPT2$. We consider a random network with 100 nodes divided into 20 clusters ($k = 20$) and three instances of random space failure scenarios with varying radius (r). In all cases, the number of reporting nodes is 10, and each case was run 10 times. The data for various netCSI approaches is given in Table 3.

Both $POS+NEG$ and NEG have a maximum search space of $\sim 10^6$ combinations in the case of $r^2 = 20$. In Figure 5(a), the average search space of $OPT1$ and $OPT2$ is 5.9% and 2.4% of the total combinations, respectively; the corresponding numbers are shown in the Table 3. Here, we considered 10 as the threshold for $OPT2$. Since size of the search space is proportional to the time complexity of netCSI, we expect $OPT2$ to have the lowest run-time of all the netCSI approaches.

In Figure 5(a), we can see that the size of the hypothesis list for $POS + NEG$ is 0.0015% of total combinations, whereas NEG lists 3.39% of total combinations as possible combinations. The reason for fewer possible combinations in the case of $POS+NEG$ when compared to NEG is because the positive information eliminates false positives. In addition, we observe that the number of possible combinations is the same for both $OPT1$ and $POS + NEG$ (0.0015%). Thus, the precision of the hypothesis list does not change, i.e., netCSI will not miss any faulty link in the ground truth and hence there is no change in accuracy with pruning. However, $OPT2$ outputs a hypothesis list with much fewer combinations (0.0009%) when compared to $POS + NEG$, so it is possible that we may not find all the faulty links in the ground truth with thresholding. Therefore, the accuracy of $OPT2$ might decrease when compared to $OPT1$ depending on the threshold value.

Algorithm	r^2	Avg. links failed	Avg total combs	Avg combs in SS	Avg possible combs	Avg CR_{GT}	Avg exact match rank	Avg $FP_{CR_{GT}}$
NEG	20	6.7	9.73x10^5	9.73x10^5	3.3x10^4	1	2.62x10^4	74.4%
	30	7.1	1.34x10^6	1.34x10^6	3.3x10^4	1	2.63x10^4	73.5%
	40	8.9	2.17x10^6	2.17x10^6	7.87x10^4	1	6.57x10^4	71.7%
OPT1	20	6.7	9.73x10^5	5.75x10^4	29.1	1	24.2	26.6%
	30	7.1	1.34x10^6	7.91x10^4	29.1	1	25.2	25.7%
	40	8.9	2.17x10^6	1.28x10^5	31.7	1	28.4	28.2%
OPT2	20	6.7	9.73x10^5	2.70x10^4	21.3	1.4	16.4	25.5%
	30	7.1	1.34x10^6	3.56x10^4	17.4	1.6	13.5	24%
	40	8.9	2.17x10^6	5.26x10^4	18.9	1.8	15.6	26.5%

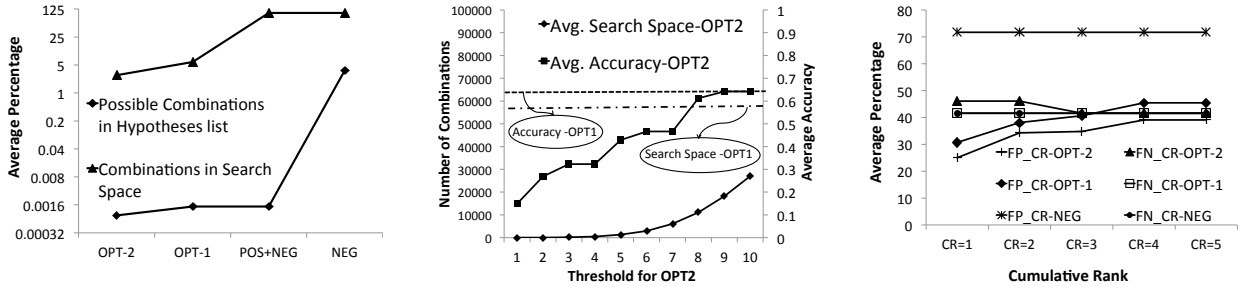
TABLE 3: Data comparing various netCSI approaches

Hence, in the case of $OPT2$, there is trade-off between the accuracy and the size of the search space. In Figure 5(b), we plot the average size of the search space and accuracy (with CR_{GT}) for different threshold values under $OPT2$. The accuracy of $OPT2$ increases with increase in the value of the threshold. For th=9, there is no loss of accuracy in $OPT2$, however the size of the search space for $OPT2$ with th=9 is much smaller than that of $OPT1$ as shown in Figure 5(b). Furthermore, in Figure 5(c), we show that both $OPT2$ with th=10 and $OPT1$ have almost the same $\%FP_{CR}$ and $\%FN_{CR}$ for different cumulative ranks. Therefore, for a properly chosen threshold value, $OPT2$ can be as effective as $OPT1$, along with the added advantage of lower search space. Henceforth, in the remainder of paper, $OPT2$ means netCSI that includes the refinements: pruning, and thresholding with a proper threshold value.

7.4.2 Performance of Ranking Algorithm

We use CFP-CI with $p_c = 0.9$ for our ranking algorithm to rate possible combinations in the hypothesis list. In Table 3, we can see that the average $CR_{GT} = 1$ for almost all netCSI approaches, i.e., we find all the faulty objects in the highest-ranked combination in the hypothesis list; however, false positives differ from one approach to another. Because of positive information, $OPT1$ (30%) and $OPT2$ (25%) have lower $\%FP_{CR_{GT}}$ compared to NEG (70.3%). Due to the smaller hypothesis list, $OPT2$ gives fewer false positives than $OPT1$. In Figure 5(c), we show that $\%FP_{CR}$ increases with increase in the cumulative

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing



(a) Precision of the hypothesis list for $r^2 = 20$

(b) Comparison between $OPT1$ and $OPT2$

(c) False Positives and False Negatives with respect to cumulative ranks

Fig. 5: Evaluation of different netCSI approaches

rank for both $OPT2$ and $OPT1$, because the CFO_{CR} list becomes bigger as we increase the cumulative rank. Please note that the average $\%FN_{CR}$ for both $OPT2$ and $OPT1$ with respect to cumulative ranks saturates at 42%.[2]

Incidentally, from Table 3 the average exact match rank of both $OPT2$ and $OPT1$ is in the *bottom* 25th percentile of the hypothesis list. This is due to the disparity in simulated failure mode (random space failure) and CFPM-CI which is employed for the ranking algorithm. Hence, most of the time the CR_{GT} lies in the *top* 5th percentile of the hypothesis list as given in Table 3. Therefore, the ranking algorithm of netCSI performs much better even though the models are distinct, which is common in practice.

7.4.3 Effect of varying number of reporting nodes

We now evaluate netCSI and compare it with MAX-COVERAGE for the situations when there is incomplete end-to-end symptom information. Since the focus is on large scale failures, acquiring end-to-end symptom information from all the reporting nodes may not be feasible. Furthermore, the major drawback in collecting all the symptom information is redundancy, which may lead to unnecessary wastage of critical resources in the network. However, with more symptoms, netCSI produces a hypothesis list that results in better accuracy and fewer false positives and false negatives.

Results are given for networks with both random and realistic topologies while varying the number of reporting nodes in Figure 6. Since the output of MAX-COVERAGE is a set of objects that could be faulty, it is straightforward to determine the accuracy. However, the hypothesis list of netCSI contains different possible combinations of objects (multiple sets), so we consider CFO_{CR} (a list of possible faulty objects) to evaluate the accuracy of netCSI. For both the networks, we employ $OPT2$ with different threshold values, and simulate a random cluster failure scenario with 10 trials.

In Figure 6(b), for realistic topology, we consider $CFO_{CR_{GT}}$ to compute the accuracy of netCSI with $OPT2$ with different threshold values. netCSI outperforms MAX-COVERAGE when the reporting nodes are varied between 10 and 100.[3] The maximum and average gain in accuracy of $OPT2_{th=7}$ are 567% (10 reporting nodes) and 128%, respectively, when compared to MAX-COVERAGE as shown in Figure 6(b). Furthermore, in Figure 6(d), we present the accuracy of $OPT2_{th=7}$ for different CR while varying the number of reporting nodes. The average gain in the accuracy for $OPT2_{th=15}$ with CR =1, 2 and 3 over MAX-COVERAGE are 107.2%, 115.8% and 119.4% respectively. Hence, the network manager can pick a very small number as CR and achieve as much accuracy as CR_{GT}.

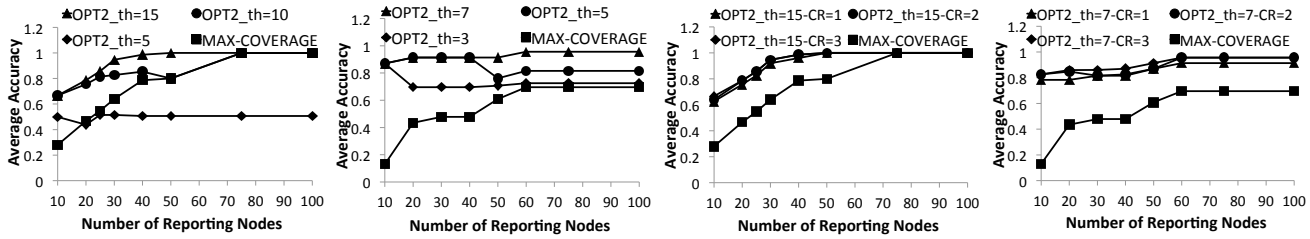
We now consider a random network of 100 nodes with 20 clusters. In Figure 6(a), we give the accuracy of $OPT2$ considering $CFO_{CR_{GT}}$. When there are fewer reporting nodes, i.e., for a number lower than 75, both $OPT2_{th=15}$ and $OPT2_{th=10}$ are more accurate than MAX-COVERAGE. Now, we specifically compare the accuracies of $OPT2_{th=15}$ and MAX-COVERAGE; the maximum gain is 138% (10 reporting nodes), and the average gain is 45%. However, $OPT2_{th=5}$ has lower accuracy than MAX-COVERAGE, and does not follow any trend as shown in Figure 6(a). The reason is that combinations in which the number of simultaneous failures is more than 5 are not considered. In Figure 6(c), the average gain in the accuracy of $OPT2_{th=15}$ over MAX-COVERAGE is almost the same for both $CR = 3$ and CR_{GT} (Figure 6(a)). Here also $OPT2_{th=15}$ with just $CR = 1$ can achieve an average gain of 41% over MAX-COVERAGE.

From the results for both the networks, we conclude that even when the network manager picks a very small number as CR, we can achieve as much accuracy as CR_{GT}. In addition, with 10 reporting nodes, we observe that $OPT2$ is more accurate for a network with a realistic topology than random topology. This is because our random topology is more dense than the considered realistic topology, and with given (small) number of symptoms netCSI outputs more possible combinations in the hypothesis list for dense networks than sparse

2. Here the number of reporting nodes are 10. By increasing the number of reporting nodes, i.e., more symptom information, we can diagnose more links that are faulty.

3. We have considered only 100 as maximum number of reporting nodes (limited no. of symptoms) and there are 315 nodes in the network, so the accuracy for all considered netCSI approaches and MAX-COVERAGE do not reach the value 1 in our results.

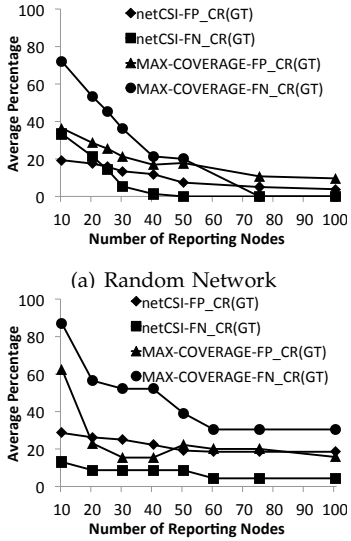
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing



(a) Random Network (b) Network with Realistic Topology (c) Random Network (d) Network with Realistic Topology

Fig. 6: Evaluation of netCSI while varying number of reporting nodes

ones.



(a) Random Network

(b) Network with realistic topology

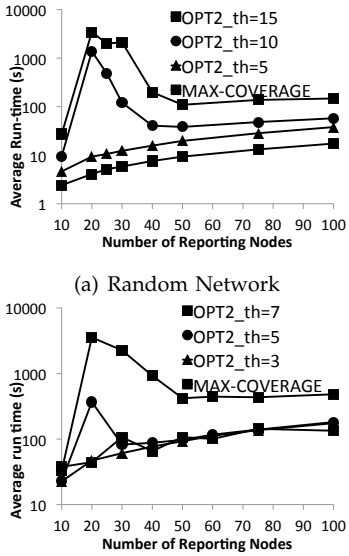
Fig. 7: False Positives and False Negatives

In Figure 7, we plot $FP_{CR_{GT}}$ and $FN_{CR_{GT}}$, with $OPT2_{th=15}$ for random network, and with $OPT2_{th=7}$ for network with realistic topology, along with false positives and false negatives of MAX-COVERAGE. In both networks, $FP_{CR_{GT}}$ and $FN_{CR_{GT}}$ of $OPT2$ are *lower* than those for MAX-COVERAGE in most of the cases. In addition, both $FP_{CR_{GT}}$ and $FN_{CR_{GT}}$ of $OPT2$ decrease with an increase in the number of reporting nodes for both the networks. However, they become more or less saturated for the cases with number of reporting nodes $>$ 50, i.e., the extra information that we get from reporting nodes 50 to 100 could be considered as redundant. Therefore, the trade-off between accuracy and overhead in collecting the information from reporting nodes, could be exploited by selecting an optimum number of reporting nodes (minimum overhead) with exactly or close to 0% false negatives, i.e., almost all the faulty objects in the ground truth are localized.

7.4.4 Run time of netCSI

In both the networks, with an increase in the value of threshold, the accuracy of $OPT2$ increases as shown in Figure 6, but at the same time, as shown in Figure 8, the run-time also increases. Hence, the gain in accuracy of $OPT2$ compared to MAX-COVERAGE comes with additional overhead of run-time. As shown in Figures 8(a) and 8(b), the run-time of $OPT2$ with different threshold values do not follow any trend when we vary the number of reporting nodes.

In Section 6, we showed that the run-time of netCSI depends on the number of possible faulty objects (N) *exponentially*, and on the number of symptoms ($c + d$) *linearly*. Even though the value of ($c + d$) increases with the increase in the number of reporting nodes, there could be an instance where N is very high in one of the trials regardless of number of reporting nodes. These type of instances can be seen in both the Figures 8(a) and 8(b) when the reporting nodes are in between 10 and 50. Moreover, since the code of netCSI is parallelized, the effect of proportionality on run-time due to number of symptoms is reduced. In addition, with more parallelization, the overall run-time of netCSI could be decreased even further, unlike MAX-COVERAGE.



(a) Random Network

(b) realistic topology

Fig. 8: Run-time of netCSI

7.4.5 netCSI with dynamic threshold value selection

As explained in Section 4.2.2, netCSI with thresholding is efficient when we select an appropriate threshold value. Until now, we evaluated netCSI with a static threshold value irrespective of the failure, i.e., we handpick a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

threshold value manually for all failures. We propose a technique called *dynamic threshold value selection* in Section 4.2.2 that selects a dynamic threshold value for each failure instance.

As discussed in Section 4.2.2, when a failure occurs, our technique uses both training data and the current observed negative symptoms to estimate the threshold value. Data is divided into different *bins* that represent different ranges of failures, i.e., in terms of number of failures. We consider the selected range as the *size of the bin*. During a failure instance we select the threshold value as the maximum number of failures in the range of the selected bin, and we call it as *dynamic threshold value*.

In Figure 9, we compared the performance of netCSI for various cases of static and dynamic threshold values. We considered static threshold values of 5, 10, 20, 30 and 40, and dynamic threshold values with bin sizes of 5, 10, 15 and 20. The number of reporting nodes is 10. We run experiments for six failure cases and show the average values of run time and accuracy in Figure 9. We consider 10,000 offline samples in the training data.

From Figure 9, we see that the accuracy of netCSI increases with increasing static threshold value, as expected. In the case of the dynamic threshold with various bin sizes, we achieve the same accuracy as the static threshold but with lower run time. We also provide the average threshold value (in parentheses in Figure 9) for all cases where the dynamic threshold value selection is used.

Even though the gap between the values of average threshold for static and dynamic cases is high, the difference in run times is just little more than significant. The main reason is our pruning technique (Section 4.2.1): many combinations with a large number of objects get pruned from the search space. Therefore, our technique enables us with an automated mechanism to choose a threshold value depending on the failure instance instead of handpicking a static value. In addition, our dynamic threshold selection can used in various other inference techniques [3].

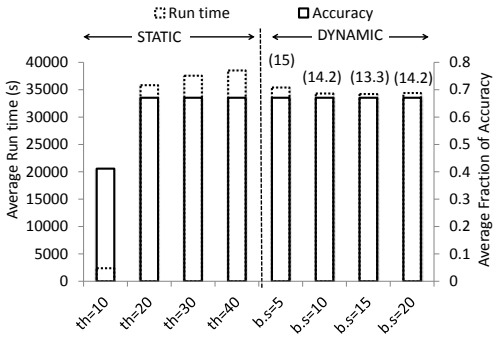


Fig. 9: Static and dynamic threshold values

In our experiments, given 10 reporting nodes, we simulated 2,000 failures that consist of both random space and random cluster failures. For two cases of training data with 10,000 and 20,000 samples, we observe that the accuracy of picking the correct bin decreases with decrease in the bin size. For 20,000 samples, we observed that 60% of the time we pick correct bins for bin sizes that are more than 15.

7.4.6 Effect of partial measurements at a reporting node

When we vary the number of reporting nodes in Section 7.4.3, we assume that complete information is available at each reporting node. However, this assumption may not be true in realistic scenarios. For example, a reporting node may not know every reachable or unreachable destination. When the reporting nodes report both negative and positive symptoms to the network manager, i.e., if the connectivity status of any destination node corresponding to a reporting node is unknown, then no symptom is reported. Therefore, the network manager receives incomplete positive and negative symptoms from various reporting nodes.

In Figure 10, we show the effect on accuracy, when the partial amount of symptoms at a selected reporting node randomly falls in the range between 30% and 90% , i.e., (30%,90%) of total number of both positive and negative symptoms. In this case, we consider 30 reporting nodes. The performance of both netCSI and MAX-COVERAGE degrade when there is partial information at reporting nodes compared to the case with complete information, as expected. However, with partial information at every reporting node, netCSI performs much better than MAX-COVERAGE.

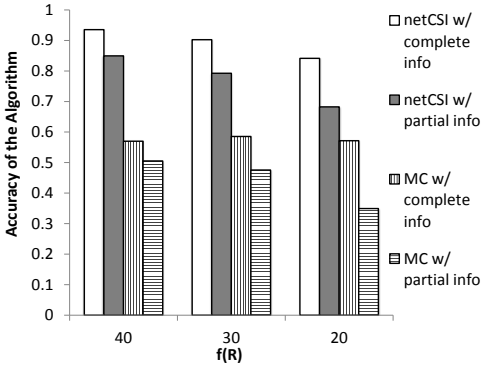


Fig. 10: Partial information at reporting nodes

We then evaluate the performance of fault diagnosis algorithms under partial information at every reporting node while varying number of reporting nodes. We also consider cases in which the amount of partial information is 90%, 60% and 30% of both positive and negative symptoms at every chosen reporting node. These results are given in Figure 11. We observe that for all cases; with 30%, 60%, 90%, and (30%,90%) partial information, the accuracy of both netCSI and MAX-COVERAGE decreases when compared to the scenario in which every reporting node has complete information. In Figure 11, please note that we do not show the results for the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

case of 60% partial information because of space constraint. With only 30% partial information, the accuracy of netCSI decreases drastically. This indicates that it is more important to choose reporting nodes with a greater amount of symptom information than selecting a large number of reporting nodes that have significantly lower amount of symptom information.

In Figure 11, MAX-COVERAGE with various amounts of partial information follows the same trend as netCSI, but performs worse than netCSI. We also observe that the accuracy of MAX-COVERAGE with complete information at every reporting node is much lower than that of netCSI under substantial partial symptom information at reporting nodes (i.e., more than 30% of complete symptom information) when there are a limited number of reporting nodes (number of reporting nodes ≤ 40) in a network of 100 nodes. This shows that netCSI is more robust than MAX-COVERAGE under partial information at a reporting node, when there are few reporting nodes.

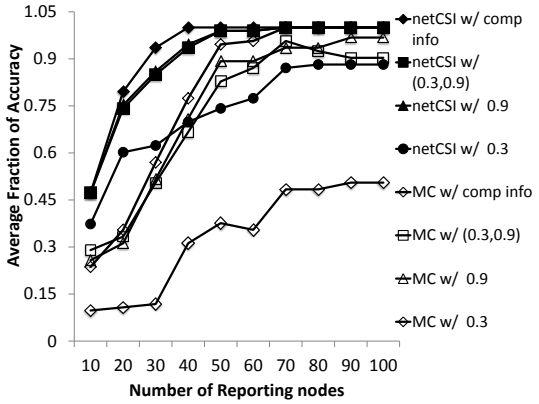


Fig. 11: Comparison of netCSI and MAX-COVERAGE under partial information at reporting nodes

7.4.7 Performance of netCSI under independent failures

We perform simulations under independent failures and observed that the hypothesis list of netCSI is highly accurate with no false negatives, but contains many false positives. For multiple runs of random independent failures, netCSI returns 70-75% of false positives. This number is very high compared to 20% of false positives returned by MAX-COVERAGE. The reason is the assumption that the geographically adjacent objects fail in clustered failures, and due to this correlation netCSI overestimates the number of failed objects. Using this insight, we propose an adaptive algorithm that works for both independent and large-scale failures in our subsequent work [25].

8 Conclusion

In this paper, we developed netCSI to accurately diagnose faults due to large scale failures in spite of having incomplete symptom information. While varying the number of reporting nodes, netCSI accomplishes an average gain of 128% and 45% in accuracy over MAX-COVERAGE for realistic and random topologies, respectively. In our results we observed that reporting nodes with many symptoms are more important than reporting nodes with very few symptoms. netCSI utilizes both positive and negative symptoms along with knowledge of the pre-failure network topology to generate a hypothesis list of possible combinations that cause the symptoms. Due to inclusion of positive symptoms, the size of the hypothesis list is reduced by almost 99.9%, and the percentage of false positives is reduced by at least 60% when compared to netCSI with only negative symptoms. In addition, the reduction in the search space of netCSI, due to pruning and thresholding, are 94% and 98%, respectively. We also show the effectiveness of dynamic threshold selector while employing *thresholding*.

Acknowledgements

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defense or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "Detection and localization of network black holes," in *Proceedings of IEEE Infocom*, 2007.

[2] S. Kandula and D. Katabi, "Shrink: A tool for failure diagnosis in ip networks," in *MineNet*, 2005.

[3] P. Bahl, R. Ch, A. Greenberg, S. K, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," in *SIGCOMM*, 2007, pp. 13–24.

[4] A. Ogielski and J. Cowie, "Internet routing behavior on 9/11 and in the following weeks," http://www.renesys.com/tech/presentations/pdf/renesys-030502-NRC-911.pdf, 2002.

[5] J. Cowie, A. Popescu, and T. Underwood, "Impact of Hurricane Katrina on Internet infrastructure," http://www.renesys.com/tech/presentations/pdf/Renesys-Katrina-Report-9sep2005.pdf, 2005.

[6] S. LaPerriere, "Taiwan Earthquake Fiber Cuts: A Service Provider View," http://www.nanog.org/meetings/nanog39/presentations/laperriere.pdf, 2007.

[7] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "Ip fault localization via risk modeling," in *ACM Symposium on Networked Systems Design & Implementation*, Berkeley, CA, USA, 2005, pp. 57–70.

[8] M. Steinder and A. S. Sethi, "Probabilistic fault diagnosis in communication systems through incremental hypothesis updating," *Comput. Netw.*, vol. 45, no. 4, pp. 537–562, 2004.

[9] A. Sahoo, K. Kant, and P. Mohapatra, "Improving BGP Convergence Delay for Large-Scale Failures," in *Dependable Systems and Networks*, 2006, pp. 323–332.

[10] B. Bassiri and S. S. Heydari, "Network survivability in large-scale regional failure scenarios," in *C3S2E*, New York, NY, USA, 2009, pp. 83–87.

[11] T. Bu, N. Duffield, F. L. Presti, and D. Towsley, "Network tomography on general topologies," *SIGMETRICS Perform. Eval. Rev.*, vol. 30, pp. 21–30, June 2002.

[12] Y. Chen, D. Bindel, H. Song, and R. H. Katz, "An algebraic approach to practical and scalable overlay network monitoring," in *SIGCOMM*, 2004, pp. 55–66.

[13] N. Duffield, "Network tomography of binary network performance characteristics," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5373– 5388, 2006.

[14] Y. Zhao, Y. Chen, and D. Bindel, "Towards unbiased end-to-end network diagnosis," in *ACM SIGCOMM*, New York, USA, 2006, pp. 219–230.

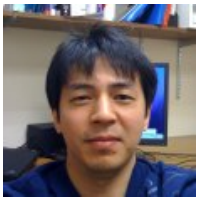
[15] V. Padmanabhan, L. Qiu, and H. Wang, "Server-based inference of internet link lossiness," in *INFOCOM*, vol. 1, 2003, pp. 145 – 155 vol.1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2014.2369051, IEEE Transactions on Dependable and Secure Computing

[16] J. Cao, D. Davis, S. V. Wiel, B. Yu, S. Vander, and W. B. Yu, "Time-varying network tomography: Router link data," *Journal of the American Statistical Association*, vol. 95, pp. 1063–1075, 2000.
[17] A. Tsang, M. Coates, and R. D. Nowak, "Network delay tomography," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2125–2136, 2003.
[18] Y. Huang, N. Feamster, and R. Teixeira, "Practical issues with using network tomography for fault diagnosis," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 53–58, 2008.
[19] D. Ghita, K. Argyraki, and P. Thiran, "Network tomography on correlated links," in *Proceedings of ACM IMC*. New York, NY, USA: ACM, 2010, pp. 225–238.
[20] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC Press, 1986.
[21] S. Tati, S. Rager, and T. La Porta, "netCSI: A Generic Fault Diagnosis Algorithm for Computer Networks," Network and Security Research Center, Penn State University, Tech. Rep. NAS-TR-0130-2010, June 2010.
[22] D. Magoni, "nem: a software for network topology analysis and modeling," in *IEEE MASCOTS 2002*, 2002, pp. 364–371.
[23] "Rocketfuel project: Internet topologies," http://www.cs.washington.edu/research/networking/rocketfuel/.
[24] B. G. Mirkin, *Mathematical Classification and Clustering*. Springer, 1996.
[25] S. Tati, B.-J. Ko, G. Cao, A. Swami, and T. L. Porta, "Adaptive algorithms for diagnosing large-scale failures in computer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, 2014.

Srikar Tati Srikar Tati is a graduate student in PhD program at Pennsylvania State University. He received the Bachelors of technology (B. Tech Hons.) degree at Indian Institute of Technology, Kharagpur in 2006 and Masters of Sciences (MS) degree at Pennsylvania State University in 2008. His research interests are in broad areas of design, modeling and analysis of network systems, large distributed systems, wireless networks etc. As part of his dissertation, he worked on various problems that arise due to challenges in failure and performance diagnosis algorithms in computer networks. He previously worked on few problems in wireless networks. He published his work in conferences such as IEEE DSN, ICNP, SRDS etc. He reviewed papers for JSAC network science, IEEE Globecom etc.

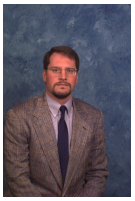
Scott Rager Scott T. Rager is a PhD candidate in the Computer Science and Engineering Department at Penn State University. He received a B.A. in Physics from the Slippery Rock University of Pennsylvania and a B.S. in Computer Engineering from Penn State both in 2009 after completing a cooperative dual-degree undergraduate program. He is presently a member of the Institute for Networking and Security Research at Penn State, and his current research interests include mobile ad hoc networks and distributed protocol designs.



Bongjun Ko Bong Jun Ko received B.S. and M.S. degrees in electrical engineering from Seoul National University, Korea, and received the Ph.D. degree in electrical engineering at Columbia University, New York. He has worked as a senior member of research staff in Philips Research North America, as a research engineer at LG Electronics in Korea, and has co-founded NeoMTel, Korea. He is currently a research staff member in IBM T. J. Watson Research Center, working in Networking Technology Area, and has research interests in modeling, analysis, and design of large distributed systems, mobile and wireless networks. He is the recipient of the best paper awards in IEEE ICNP 2003 and in IEEE/IFIP IM 2013.

Guohong Cao Guohong Cao received the BS degree in computer science from Xian Jiaotong University and received the PhD degree in computer science from the Ohio State University in 1999. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a Professor. He has published more than 200 papers in the areas of wireless networks, wireless security, vehicular networks, wireless sensor networks, cache management, and distributed fault tolerant computing. He has served on the editorial board of IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, IEEE Transactions on Vehicular Technology, and has served on the organizing and technical program committees of many conferences, including the TPC Chair/Co-Chair of IEEE SRDS'2009, MASS'2010, and INFOCOM'2013. He was a recipient of the NSF CAREER award in 2001. He is a Fellow of the IEEE.

Ananthram Swami Ananthram Swami (F'08) received the B.Tech. degree from IIT-Bombay; the M.S. degree from Rice University, and the Ph.D. degree from the University of Southern California (USC), all in Electrical Engineering. He has held positions with Unocal Corporation, USC, CS-3 and Malgudi Systems. He was a Statistical Consultant to the California Lottery, developed a Matlab-based toolbox for non-Gaussian signal processing, and has held visiting faculty positions at INP, Toulouse. He is with the US Army Research Laboratory (ARL) where he is the Army's ST for Network Science. His work is in the broad area of network science, with emphasis on wireless communication networks. He is an ARL Fellow and Fellow of the IEEE.



Thomas F. La Porta Thomas F. La Porta is the William E. Leonhard Chair Professor in the Computer Science and Engineering Department at Penn State. He received his B.S.E.E. and M.S.E.E. degrees from The Cooper Union, New York, NY, and his Ph.D. degree in Electrical Engineering from Columbia University, New York, NY. He joined Penn State in 2002. He is the Director of the Institute of Networking and Security Research at Penn State. Prior to joining Penn State, Dr. La Porta was with Bell Laboratories since 1986. He was the Director of the Mobile Networking Research Department in Bell Laboratories, Lucent Technologies. He is an IEEE Fellow, Bell Labs Fellow, received the Bell Labs Distinguished Technical Staff Award in 1996, and an Eta Kappa Nu Outstanding Young Electrical Engineer Award in 1996. He also won a Thomas Alva Edison Patent Awards in 2005 and 2009. Dr. La Porta was the founding Editor-in-Chief of the IEEE Transactions on Mobile Computing, and currently serves on its Steering Committee (he was Chair from 2009-2010). He served as Editor-in-Chief of IEEE Personal Communications Magazine. He was the Director of Magazines for the IEEE Communications Society and was on its Board of Governors for three years. He has published numerous papers and holds 35 patents.