Efficient Probing Techniques for Fault Diagnosis [1]

Maitreya Natu
Dept. of Computer and Information Sciences
University of Delaware
Newark, DE, USA, 19716
Email: natu@cis.udel.edu

Adarshpal S. Sethi
Dept. of Computer and Information Sciences
University of Delaware
Newark, DE, USA, 19716
Email: sethi@cis.udel.edu

Abstract— **Increase in the network usage and the widespread application of networks for more and more performance critical applications has caused a demand for tools that can monitor network health with minimum management traffic. Adaptive probing holds a potential to provide effective tools for end-to-end monitoring and fault diagnosis over a network. In this paper we present adaptive probing tools that meet the requirements to provide an effective and efficient solution for fault diagnosis. In this paper, we propose adaptive probing based algorithms to perform fault localization by adapting the probe set to localize the faults in the network. We compare the performance and efficiency of the proposed algorithms through simulation results.**

Index Terms— **Adaptive probing, Probe selection, Network monitoring, Failure detection, Fault localization.**

I. Introduction

With increasing number of business applications relying on the network infrastructure, it becomes important to provide an efficient mechanism to monitor the network for availability, connectivity and performance. One important application of network monitoring is to detect the presence of faults in a network. The nature of faults vary based on the domain of deployment. For instance, faults could be at the network level (node and link failures), or at application level (failure of a database server or application server). Targeted failures could be fail-stop or Byzantine in nature.

A promising approach to probing is *adaptive probing* where the probe set is adapted to the observed network conditions [2]. First a small set of probes is sent for *failure detection*. This probe set is capable only of detecting any failure in the network. Note that these probes might not be able to localize the exact failure, but they can only detect a failure in the network. Additional probes can then be sent for *fault localization* on the selected area of interest in the network to perform further diagnosis. The additional probes are adapted to the observed network conditions by using the interactive approach of analyzing the previous probe results to narrow down the problem areas and selecting new probes to pin-point failure locations using a fewer number of probes.

Figure 1 shows the architecture for an adaptive probing system for fault localization with three main components. The *probe station selection* module selects the best locations to deploy the probe stations using the available dependency model information about the network routes. We have discussed the probe station selection problem in [10]. The *failure detection* component selects and periodically sends the smallest set of probes from the available probes, which can be used to detect a failure in the managed network. The failure detection module triggers fault localization when a failure is detected. We have discussed the failure detection problem in [11]. The *fault localization* module infers the network state from the observed probe results and the probe's dependency information. It then selects additional probes online to obtain more information for localizing the fault. It repeats this process of analysis and selection till the fault localization is complete. In this paper, we discuss the problem of probe selection for fault localization.

Probe set selection criteria for failure detection and fault localization is different. Probe set for failure detection is selected such that all network elements are probed. However fault localization requires a probe set that uniquely diagnoses the suspected network elements. Probes for failure detection are sent periodically and thus the management traffic produced should be low enough that it does not affect the performance of other applications. Moreover the time constraints on probe set selection for failure detection are less stringent than that for fault localization. Fault localization is done only when some problem is encountered. Thus probes for fault localization should be selected such that the fault localization can be done in minimum amount of time and at the same time the network in the identified problem areas should not be overwhlemed with the management traffic.

In this paper, we present algorithms to select probes to perform fault localization. The algorithms select additional probes that need to be sent over the network to perform a deeper diagnosis once a failure is detected. We first present two algorithms called Min-Search and Max-Search that use a Greedy approach to select probes. We then present a Binary-Search algorithm that sends out probes on the failed probe paths in a binary search fashion to detect health of nodes on these paths. We present simulation results for experimental evaluation of the three algorithms.

[1]Prepared through collaborative participation in the Communications and Networks Consortiumsponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

[2]Some researchers [4] Have used the term *active probing* to refer to this type of probing

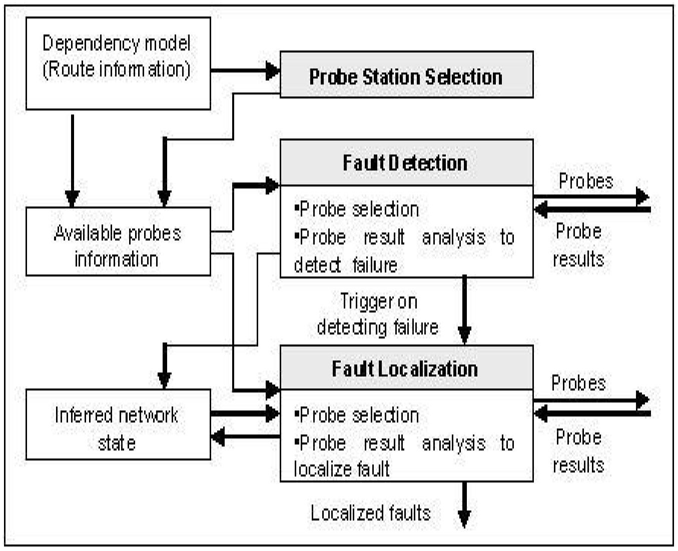
Dependency model (Route information)
Probe Station Selection
Fault Detection
•Probe selection
•Probe result analysis to detect failure
Probes
Probe results
Available probes information
Trigger on detecting failure
Fault Localization
•Probe selection
•Probe result analysis to localize fault
Probes
Probe results
Inferred network state
Localized faults

Fig. 1. System architecture for failure diagnosis

A. Related work

Various approaches have been proposed in the past for network monitoring in order to measure different performance metrics [14], [1], [7], [4]. Due to space reasons we survey only those apporoaches that directly relate to probe selection.

The probing tools proposed in the past, consist of connectivity, latency and bandwidth measurement tools such as pathchar[7], Skitter [5], pathrate [6], PathChirp [12], NetTimer [8] etc. These tools measure end-to-end network performance metrics by sending probes in the network. For instance, PathChirp [12] is an adaptive probing tool for estimating available bandwidth on a communication network path. Skitter [5] sends a sequence of probes to a set of destinations and measures the latency of a link as the difference in round-trip times of the two probes to link end-points. The end-to-end nature of probes allows measurement of end-to-end performance metrics involving several components [9], [2]. These tools however send management traffic into the network. Thus care needs to be taken to optimize the probes that are being sent out in the network, so that the network operation is not affected by the management traffic.

Network probing with low overhead has prompted development of new monitoring approaches. Rish et. al. [3], [4] proposed some heuristic based approaches to be used for both probe set selection for failure detection and fault localization. Subtractive search proposed in [3] is fast but its effectiveness in finding the minimal set depends on the order in which probes are explored. Another approach proposed in [3] is additive search, where at each step the probe giving the most informative decomposition is added to the probe set. In [11], we have presented a Greedy algorithm for probe selection for failure detection and presented a comparison of the additive search [3] and the proposed Greedy search. An adaptive probing approach is proposed in [13] to select probes for fault detection by incrementally selecting probes that cover the nodes that are not yet covered.

II. Failure detection

In this section, we introduce the probe selection algorithm to perform failure detection. We have presented this algorithm in [11] and include it here for the sake of completeness. We use this algorithm together with the proposed fault localization algorithms to develop a complete fault diagnosis tool.

Probes for failure detection need to be sent from the probe stations at different network locations to monitor the health of the managed network components. These probes should be selected such that, in the presence of a fault in the network, some of the probes should fail, causing the failure to be detected by the manager. As the management traffic for failure detection runs periodically, it should be optimized to prevent overwhelming the network resources and affecting the performance of other applications using the network. Thus the aim is to find a minimal set of probes that can detect the health of all the managed components such that if any component fails, at least one of the probes should report failure. In this section, we present an algorithm for optimizing the selection of probes for detecting the presence of a fault in the network.

A. Probe set selection criteria

Different nodes are probed by a different number of probes, depending on the routes used. Nodes that are probed by a less number of probes narrow down the search space for probe selection. Consider the case where a node n is probed by only one probe. In this case, the only probe probing node n must always be selected, irrespective of the number of nodes it covers. Consider another case, where only two probes pass through a node n. Then one of the two probes must be selected to cover node n. In this situation, the probe covering a larger number of uncovered nodes is the better choice.

Algorithm GFD describes a Greedy approximation algorithm that explores the information contained in the dependencies between probes and network components. The algorithm selects the network element which is probed by the least number of probes, using the dependency information between probes and probed elements. Out of all the probes probing element n, the algorithm selects the one which goes through maximum number of nodes that are not yet probed.

III. Fault localization

The fault localization process is triggered when a failure is detected by the failure detection probes. Failure of some of the probes sent for failure detection indicates the presence of one or more faults over the failed probe paths. However these probes might not be able to locate the exact cause of failure. Hence additional probes must be sent over the identified problem areas to localize the fault. In this section, we present various approaches to select additional probes for localization to a finer granularity.

For each failed probe, the fault localization module sends additional probes over the nodes on the failed probe paths. The probes need to be sent such that health of all the nodes on the failed probe paths can be determined. However the presence of failures makes certain nodes unreachable from certain probe

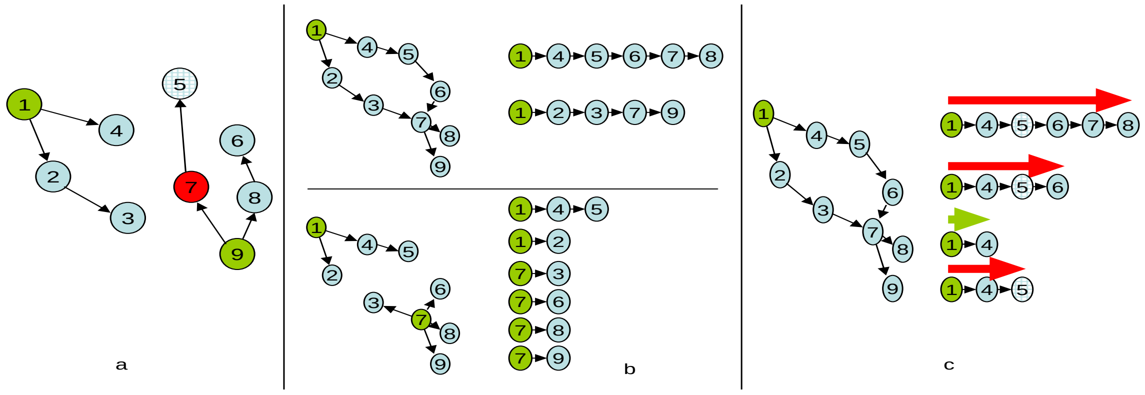

Fig. 2. (a.)Failure of node 7 makes node 5 unreachable from probe station node 9; (b.) Network nodes probed by a small set of long probes or a large set of short probes; (c.)Probes sent in a Binary search fashion on path 1− > 8 to detect failure at node 5;

```
Algorithm GFD: Greedy algorithm for probe set selection for failure detection
 1: Inputs:
 2: N: The set of nodes. PS: The set of probe stations.
 3: AvailableProbes: Set of probes that can be sent from probe stations to
    other nodes in the network.
 4: Initialize a set NonProbedNodes to all non-probe station nodes, i.e., N -
    PS
 5: Select the node m from NonProbedNodes, that is probed by least
    number of probes
 6: Out of all the probes in AvailableProbes probing node m, select the
    probe q that probes largest number of NonProbedNodes
 7: Remove all nodes probed by probe q from NonProbedNodes
 8: Remove probe q from AvailableProbes
 9: Add probe q to SelectedProbes
10: if ((NonProbedNodes $\neq$ Null) & (AvailableProbes $\neq$ Null)) then
11:     Repeat steps 5 - 10
12: end if
13: if NonProbedNodes = Null then
14:     Return the SelectedProbes and exit
15: end if
16: if AvailableProbes = Null then
17:     Report "Insufficient probes" and exit
18: end if
```

paths. This can be explained from the example shown in Figure 2a. Figure 2a shows probe paths from probe stations 1 and 9 to other nodes in the network. Consider the probe path 9→7→5. Failure of node 7 makes node 5 unreachable for the probe station node 9. Also, as certain failures are identified, probe paths need to be selected to probe the rest of the nodes such that the probe paths do not pass through the already identified failed nodes. In this section we present various approaches to select such probes.

A. *Greedy algorithm*

In this section we present algorithms based on the Greedy approach to build a probe set for fault localization. We first present Algorithm NM that describes the operations performed by the network manager. The network manager first sends the probe set for failure detection. If no failure is observed on these probes, the manager waits for some time and then resends the same probe set for failure detection. However, if a failure is observed, the manager performs deeper diagnosis of failed probe paths by calling the Algorithm GFL (Greedy Fault Localization) algorithm. Algorithm GFL receives the information about the passed and failed probes and the information already collected about the passed and failed nodes. The algorithm updates the passed and failed nodes by analyzing the received information. The algorithm then computes a set of probes for fault localization to be sent for further analysis of nodes whose health is not yet determined. The network manager sends these additional probes and repeats the same process till health of all nodes is determined.

Algorithm GFL presents the Greedy algorithm to select probes for fault localization. The algorithm maintains sets of failed, passed, and suspected nodes. The sets of passed and failed nodes respectively contain the successes and failures that are identified by the algorithm. The set of suspected nodes contains the nodes whose health needs to be determined. This suspected node set is initialized to all nodes that are present on the failed probe paths. The nodes lying on the paths of successful probes are removed from the set of suspected nodes and are added to the set of passed nodes. If a failed probe path contains only one suspected node and all other nodes on that path belong to the passed nodes set then the suspected node is declared as a failed node and is added to the failed nodes set. The algorithm then builds a probe set to be sent over the network to determine the health of the suspected nodes. The success and failure of the additional probes sent affect the set of suspected, failed, and passed nodes.

We present two approaches to select the probes for probing the nodes in the suspected node set. One approach is to iteratively select a probe that covers maximum number of suspected nodes. The success of such a probe gives a large amount of information by removing all the nodes on that probe path from the suspected node set. However if the probe fails then the probe does not give much information to significantly narrow down the search space of a failed node. Hence another approach could be to select a probe for each suspected node such that it goes through the least number of other suspected nodes. The success of such a probe gives the information about good health of a small number of nodes, reducing the suspected node set only by a small amount. However, the failure of such a probe narrows down the search space significantly. Figure 2b shows an example of how a network can be probed by a set of long or short probes. Success of probe 1→8 gives information about good health of nodes 4, 5, 6, 7, and 8, while its failure narrows down the failure to a set of 5 nodes. This set would need more probes for further diagnosis. On the other hand, success of a smaller probe 1→2 gives very little information indicating good health of the single node 2, but failure of this probe narrows the fault localization to a single node, node 2. This set requires no further fault localization.

Based on this concept we present two algorithms to select probes for fault localization. The basic algorithm presented in Algorithm GFL stays the same for the two approaches. The two approaches differ in the probe selection function. The function SelectFLProbes() in Algorithm GFL selects the set of probes from the set of available probes to diagnose the suspected nodes. We present the two approaches to select this probe set, namely Max Search and Min Search. Thus we present two different implementations of the SelectFLProbes function based on the Max search and Min search to present a Greedy Fault Localization Algorithm with Max search and Min Search.

Algorithm NM: Network Monitoring Algorithm (Algorithm to send probes for monitoring network health and localizing causes of failure)

```
   Input  : NetworkNodes, ProbeSet
   Output: FailedNodes
 1 FDProbes ← Failure Detection Probes (from Algorithm 1);
 2 while no failure do
 3     Send FDProbes; Wait for t time units;
 4 end
 5 Identify the PassedProbes, FailedProbes;
 6 FailedNodes ← NULL; PassedNodes ← NULL;
 7 while |FailedProbes| ≠ NULL do
 8     (FLProbes, FailedNodes, PassedNodes) ← GFL(FailedProbes,
       PassedProbes, FailedNodes, PassedNodes, ProbeSet);
 9     Send FLProbes and identify PassedProbes and FailedProbes;
10     Remove probes ∈ FLProbes from ProbeSet;
11 end
12 Return FailedNodes and exit;
```

Algorithm GFL: Greedy Fault Localization Algorithm (Probe selection for fault localization in the network)

```
   Input  : FailedProbes, PassedProbes, FailedNodes, PassedNodes,
            ProbeSet
   Output: Probe set for further fault localization, FailedNodes,
           PassedNodes
 1 Add nodes on the passed probe paths to PassedNodes;
 2 Initialize a set SuspectedNodes to nodes that lie on failed probe paths
   and are not present in the PassedNodes set and are not the
   ProbeStationNodes;
 3 foreach probe p ∈ FailedProbes do
 4     PathSuspectedNodes ← ProbePathNodes(p) ∩ SuspectedNodes;
 5     if |PathSuspectedNodes| = 1 then
 6         add PathSuspectedNodes to the FailedNodes;
 7     end
 8 end
 9 Remove nodes ∈ FailedNodes from SuspectedNodes;
10 Build a set AvailableProbes to the probes from ProbeSet that pass
   through SuspectedNodes and do not pass through FailedNodes;
11 FLProbes ← SelectFLProbes (SuspectedNodes, AvailableProbes);
12 return FLProbes, FailedNodes, PassedNodes;
```

1) Max search: As explained above, the Max Search approach selects a probe that covers maximum number of suspected nodes. In this implementation, the procedure SelectFLProbes() returns a set of probes from the available probes by iteratively selecting probes that cover maximum number of uncovered nodes till all the suspected nodes are covered.

Procedure `SelectFLProbes` (*SuspectedNodes, AvailableProbes*) {Probe selection using Max search}

```
begin
    LocalizationProbes ← NULL;
    TargetNodes ← SuspectedNodes;
    ProbeSpace ← AvailableProbes;
    while |TargetNodes| ≠ NULL do
        NextProbe ← Probe ∈ ProbeSpace that covers maximum
        number of TargetNodes;
        Add NextProbe to LocalizationProbes;
        Remove NextProbe from ProbeSpace;
        Remove ProbedNodes(NextProbe) from TargetNodes;
    end
    return LocalizationProbes;
end
```

2) Min search: The Min Search approach works on the concept of selecting a probe for each suspected node set such that the selected probe goes through minimum number of other nodes in the suspected node set. For implementing the Min search approach, the procedure SelectFLProbes() returns a set of probes for fault localization using this approach.

Procedure `SelectFLProbes` (*SuspectedNodes, AvailableProbes*) {Probe selection using Min search}

```
LocalizationProbes ← NULL;
TargetNodes ← SuspectedNodes;
ProbeSpace ← AvailableProbes;
foreach node n ∈ TargetNodes do
    NextProbe ← Probe that passes through node n and through
    minimum number of other TargetNodes;
    Add NextProbe to the LocalizationProbes;
    Remove NextProbe from ProbeSpace;
    Remove ProbedNodes(NextProbe) from TargetNodes;
end
Return LocalizationProbes;
```

B. Binary search

In this section, we present another approach for selecting probes for fault localization. Here we propose to diagnose each failed probe path independently. On each failed probe path, additional probes are sent till one failure on that path is diagnosed. These probes are sent in a binary search fashion. On a failed probe path, a probe is first sent from the probe station half way on the probe path. If this probe fails, further diagnosis is done on the first half of the probe path. On the other hand, if this probe succeeds, then the later half of the probe path is diagnosed in similar fashion. Figure 2c shows an example of how probes are sent in a binary search fashion to identify a failed node on the probe path. In this figure, probes are sent from probe station node 1. Consider that node 5 has failed and that the failure was detected on observing a failure of probe 1→8. Binary search probe selection then sends probe 1→6. On observing a failure on this path, the search area is reduced to the first half of the probe path focusing on nodes

1, 4, 5, and 6. Continuing probe selection in the binary search fashion, probe 1→4 is sent. Success of this probe indicates good health of nodes 1 and 4, leaving nodes 5 and 6 as the suspected nodes. Next a probe is sent from node 1 to node 5. Failure of this probe indicates a failure of node 5.

This process identifies one failed node on each probe path. On each of these probe paths the health of the nodes behind the identified failed node might still be unknown. Thus a suspected node set is again created that consists of the unidentified nodes that lie behind the failed nodes on the probe paths. The nodes that are already known to be in good health or failed, are removed from this set. The similar fault localization process is repeated for this newly formed suspected node set. The process is repeated till health of all the nodes is determined. Algorithm BSFL presents the proposed approach.

Algorithm BSFL: Binary Search Fault Localization Algorithm (Probe selection for fault localization in the network)

```
   input  : Nodes, ProbeSet
   output: FailedNodes
1  FDProbes = FDetection (Nodes, ProbeSet) (from Algorithm 1);
2  Send FDProbes;
3  if no probe failed then
4      Declare FailedNodes and return;
5  end
6  SuspectedNodes ← NULL;
7  foreach failed probe p = ( $n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_m$ ) do
8      n ← BinarySearch ($n_1, n_1, n_m$);
9      Add n to FailedNodes;
10     Add all nodes following n on the path ($n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_m$)
       to SuspectedNodes ;
11     Remove probes passing through node n from the ProbeSet;
12 end
13 BSFL (SuspectedNodes, ProbeSet);
```

Procedure `BinarySearch` (*StationNode, StartNodePosition, EndNodePosition*)

```
if StartNodePosition = EndNodePosition then
    return StartNodePosition;
end
TargetPosition ← ⌈ (StartNodePosition + EndNodePosition)/2 ⌉;
Send probe from StationNode to node at position TargetPosition;
if probe fails then
    BinarySearch (StationNode, StartNodePosition, TargetPosition);
end
else
    BinarySearch (StationNode, TargetPosition, EndNodePosition);
end
```

IV. Simulation results

In this section we present experimental evaluation of the probe selection algorithms for fault localization introduced in this paper. We compare the performance of the three algorithms by computing the number of probes required by the algorithms to localize a failure. Along with the number of probes, we also compute the time required by the three algorithms in performing the localization.

A. Simulation model

We simulated various network topologies varying in the network sizes and average node degrees. Let MD, AD, and N represent the maximum node degree, average node degree and number of nodes in the network. Given these three parameters, we create a network of N nodes, randomly introducing N*AD links such that no node has a network degree greater than MD, and also ensuring that the network is connected. We conducted experiments on network sizes ranging from 10 to 50 nodes with average network degrees ranging from 3 to 6 and maximum node degree set to min(20, network size) . Each point plotted on the graph is an average of 20 runs.

In each experiment we first ran a probe station selection algorithm presented in [10]. This identifies a set of nodes as probe stations from where probes can be sent to monitor the network. We used the probes that can be sent from these probe stations to build the probe set. We randomly introduced three node failures and ran Algorithm BSFL and Algorithm NM using Min Search and Max Search approaches for probe selection. We calculated the number of probes required by the three algorithms and the time required by the three algorithms in terms of probe trip times. We consider the average time taken to send a set of probes and get back the probe results as one probe trip time. Using this metric, we compare the time taken by the three algorithms in localizing the failures.

Figure 3 presents graphs for the probe set size computed and the time taken by the Min search, Max search and the Binary search fault localization approaches. Figure 3 shows that the Min search approach performs better than the Binary search and Max search approaches in terms of the number of probes required to perform fault localization. Both Max search and Binary search localize the fault in almost same number of probes. Min search selects probes of small path length to analyze the suspected nodes. These small length probes are able to infer the exact health of the suspected node in very few iterations because of less number of nodes involved, thereby reducing the number of probes needed for further diagnosis. Max search requires more number of probes for localization because the large length probes are more likely to fail due to presence of failed nodes. A failed large probe gives less information than a failed short probe. This results in a need of more probes to perform diagnosis. Binary search uses the failure detection algorithm to compute the probe set for sending probes over suspected nodes. This algorithm like Max search tries to minimize the number of probes required to cover all suspected nodes. A Binary search treats each probe path independently requiring log(n) probes for analyzing a probe of length n and to identify a single failure on each probe path. Binary search approach would need additional probes to identify the health of unknown nodes behind a failed node on each failed probe path.

Figure 3 also compares the time required by the three algorithms in localizing the failure where the time is measured in terms of probe trips. We observe that Min search and Binary search take less time while Max search takes more diagnosis

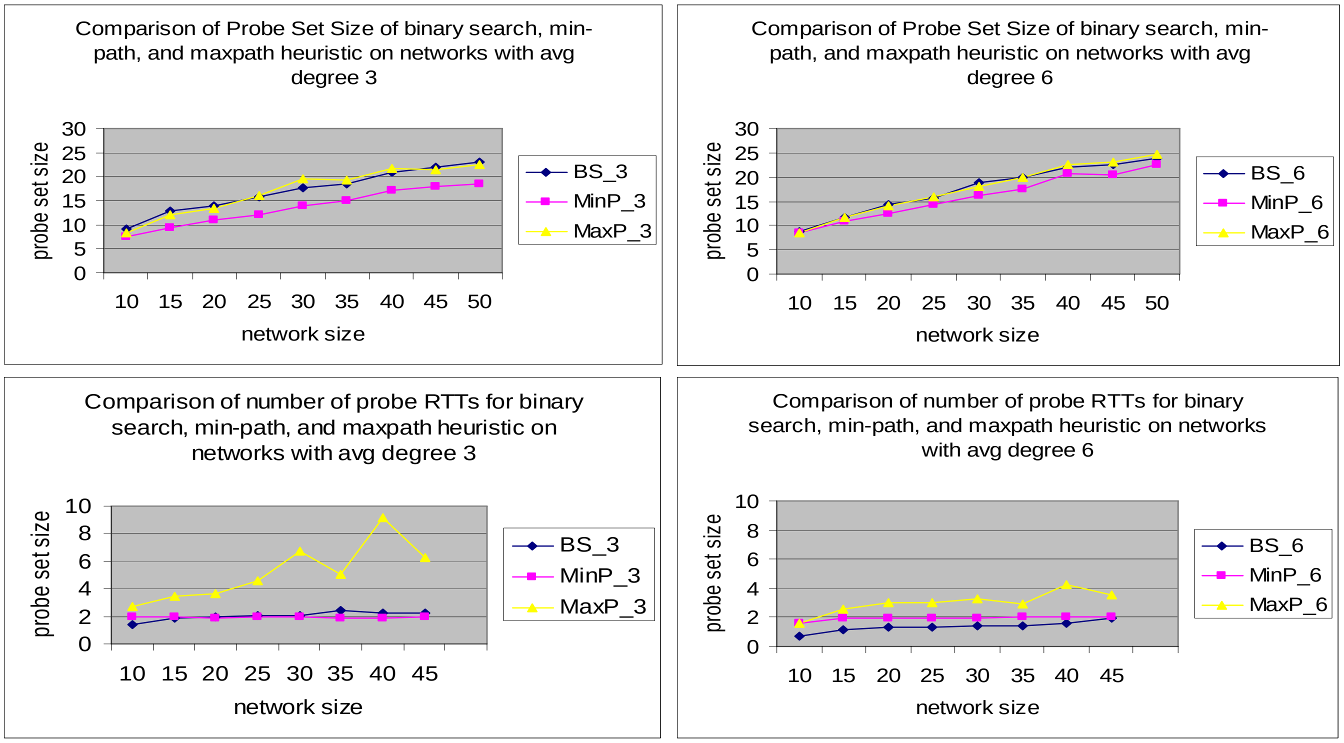

Fig. 3. Comparison of the probe set size and time computed by Min, Max and Binary search fault localization approaches on networks with different sizes and with varying average node degrees

time. Performance of Binary search is close to Min search. Min search diagnoses all suspected nodes in parallel. Moreover, the small probe size leads to quicker diagnosis of a node health. Max search uses probes of larger lengths to cover all suspected nodes. This leads to increased number of iterations because failure of a long probe gives less information and a large probe has a higher probability of failure. This leads to an increased number of iterations in localizing the faults. Binary search though uses almost as many probes as Max search but operates in parallel on all probe paths leading to a faster diagnosis.

V. Conclusion and Future work

In this paper, we presented an adaptive probing based approach to perform fault localization in networks. We proposed fault localization algorithms to adapt the probe set to the observed network conditions and send probes in an interactive manner to perform the diagnosis. In the past, we have presented an algorithm for probe selection to detect a failure in the network. In this paper, we presented fault localization algorithms and demonstrated how these algorithms can be used with the failure detection module to build a fault diagnosis tool. We presented three algorithms to select probes for fault localization in an adaptive manner and presented the performance evaluation of these algorithms. The end-to-end nature of probes and optimized traffic overhead makes adaptive probing a promising tool for various monitoring applications.

As a part of future research, we aim to consider node mobility and scenarios with incomplete or inaccurate dependency information while selecting probes for fault localization. We also aim to develop a probe station communication protocol and build a decentralized fault localization system.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government.

References

[1] Netflow services and applications, Cisco Systems, 1999.
[2] Y. Bejerano and Rajeev Rastogi. Robust monitoring of link delays and faults in IP networks. In *IEEE INFOCOM, San Francisco, CA*, Mar 2003.
[3] M. Brodie, I. Rish, and S. Ma. Optimizing probe selection for fault localization. In *Distributed Systems Operations Management*, pages 1147–1157, 2001.
[4] M. Brodie, I. Rish, S. Ma, G. Grabarnik, and N. Odintsova. Active probing. Technical report, IBM, 2002.
[5] R. L. Carter and M. E. Crovella. Server selection using dynamic path characterization in wide-area networks. In *IEEE INFOCOM'99, Kobe, Japan*, Apr 1997.
[6] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *IEEE INFOCOM'01, Alaska*, Apr 2001.
[7] A. B. Downey. Using pathchar to estimate internet link characteristics. In *ACM SIGCOMM, Cambridge, MA*, 1999.
[8] K. Lai and M. Baker. Measuring bandwidth. In *IEEE INFOCOM'99, New York City, NY*, Mar 1999.
[9] F. Li and M. Thottan. End-to-end service quality measurement using source-routed probes. In *IEEE INFOCOM, Barcelona, Spain*, Apr 2006.
[10] M. Natu and A. S. Sethi. Probe station placement for robust monitoring of networks. *Submitted to Journal of Network and Systems Management*.
[11] M. Natu and A. S. Sethi. Active probing approach for fault localization in computer networks. In *E2EMON'06, Vancouver, Canada*, 2006.
[12] V.J. Ribeiro, R.H. Riedi, R.G. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient available bandwidth estimation for network paths. In *Passive and Active Measurement Workshop*, 2003.
[13] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez. Adaptive diagnosis in distributed systems. *IEEE Transactions on Neural Networks*, 6(5):1088–1109, Sep. 2005.
[14] W. Stallings. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley Longman Inc., 3 edition, 1999.