Router Group Monitoring: Making Traffic Trajectory Error Detection More Efficient

Bo Zhang Guohui Wang Angela Yun Zhu T. S. Eugene Ng
Department of Computer Science
Rice University

***Abstract*—Detecting errors in traffic trajectories (i.e., packet forwarding paths) is important to operational networks. Several different traffic monitoring algorithms such as Trajectory Sampling, PSAMP, and Fatih can be used for traffic trajectory error detection. However, a straight-forward application of these algorithms will incur the overhead of simultaneously monitoring all network interfaces in a network for the packets of interest. In this paper, we propose a novel technique called router group monitoring to improve the efficiency of trajectory error detection by only monitoring the periphery interfaces of a set of selected router groups. We analyze a large number of real network topologies and show that effective router groups with high trajectory error detection rates exist in all cases. However, for router group monitoring to be practical, those effective router groups must be identified efficiently. To this end, we develop an analytical model for quickly and accurately estimating the detection rates of different router groups. Based on this model, we propose an algorithm to select a set of router groups that can achieve complete error detection and low monitoring overhead. Finally, we show that the router group monitoring technique can significantly improve the efficiency of trajectory error detection based on Trajectory Sampling or Fatih.**

***Index Terms*—Traffic trajectory error, monitoring, sampling, detection, router group**

I. Introduction

Routers are complex systems, so they are prone to implementation bugs. Publicly available bug reports for Cisco routers and open-source router Quagga [4] show that a large number of router bugs, once triggered, can cause various *traffic trajectory errors* including forwarding error (i.e., traffic deviating from its intended forwarding paths), dropping error (i.e., traffic being mistakenly dropped) and filter-bypass error (i.e., unauthorized traffic bypassing packet filters). These traffic trajectory errors are serious problems because they may cause network applications to fail and create security loopholes for network intruders to exploit.

A. The Need For Detecting Traffic Trajectory Errors

Here we list some recently reported Quagga and Cisco router bugs that can cause traffic trajectory errors:

For Quagga routing software, multiple reported bugs can result in incorrect routing tables such as new routes being ignored (Quagga Bugzilla [3] bug ID: 298, 464, 518), expired routes being used (Quagga Bugzilla bug ID: 85, 134), incorrect routes being installed (Quagga Bugzilla bug ID: 238, 546), and routers stopping adapting to topology change (Quagga Bugzilla bug ID: 107). For Cisco routers, multiple reported bugs can cause a network interface to drop all future packets (Cisco Advisory IDs [1]: cisco-sa-20080326-IPv4IPv6, cisco-sa-20090325-udp). Another bug may cause the firewall module of Cisco routers to stop forwarding traffic (Advisory ID: cisco-sa-20090819-fwsm). Another bug may change the forwarding table of a router (Advisory ID: cisco-sa-20080326-mvpn). Yet another bug may invalidate control-plane access control lists (Advisory ID: cisco-sa-20080604-asa). Multiple bugs can stop access control list from working, so that unauthorized traffic can go through the affected routers (Advisory IDs: cisco-sa-20090923-acl, cisco-sa-20071017-fwsm, cisco-sa-20011114-gsr-acl, cisco-sa-20000803-grs-acl-bypass-dos). Another bug (Advisory ID: cisco-sa-20070412-wlc) could allow packet filters to be inserted so that some packets may be dropped silently.

According to Cisco advisory [1] and Quagga Bugzilla [3], the reported Cisco and Quagga router bugs exist in multiple versions of Cisco IOS and Quagga routing software, that is, many deployed routers may be affected by those bugs. Worse, there are likely many more bugs yet to be discovered.

Eliminating router implementation bugs during development is hard, because no vendor can test all network designs, configurations and traffic patterns that can exist in the real world. Note that static router configuration correctness checking tools [13][12] or control plane monitoring mechanisms [24] do not help here. This is because the bugs may exist even when routers are correctly configured by the operator, and the control plane (e.g. OSPF, BGP) of a buggy router may continue to appear to be working correctly. Therefore, it would be very beneficial for the network operator to have the ability to detect traffic trajectory errors quickly and efficiently when they are eventually triggered in the field.

B. Building Blocks of Trajectory Error Detection Systems

A number of traffic trajectory error detection systems (e.g.,WATCHERS [7][15], Fatih [20][21], SATS [19], Trajectory Sampling [10]) have been proposed. The detailed description of these systems is in Section VI. Although the proposed systems employ different approaches to detecting trajectory errors, they share two common building blocks:

- **Tracking routers' control states:** The control states (e.g. forwarding table, packet filters) of a router determine its intended behaviors, i.e., how it should process packets. Commercial software such as Packet Design's Route Explorer [22] can accurately track the routing states of multiple routing protocols including BGP, OSPF, IS-IS, and EIGRP. Existing trajectory error detection systems employ their own approaches to tracking routers' control states. Our trajectory error detection prototype [30] can also track the BGP and OSPF states by passively collecting BGP and OSPF messages. These existing implementations suggest that it is feasible to accurately track routers' control states. Therefore, this paper will focus on the other orthogonal building block: monitoring the actual traffic trajectories.

- **Monitoring the actual traffic trajectories:** To monitor how the traffic flows through a network, the existing techniques need to enable the traffic monitoring function on *all* the network interfaces in a network. Ideally, each interface should monitor all traffic that is going through it. However, monitoring all traffic at full rate will incur a high monitoring and reporting overhead on both routers and the network, so in practice only a certain fraction of traffic is sampled for each monitoring period. Specifically, during each monitoring period, all monitoring devices deterministically choose a certain subset of the packets to be sampled by all interfaces. Different subsets of packets are then monitored during different monitoring periods. Once the actual traffic trajectories are obtained, traffic trajectory errors can then be detected by comparing the observed traffic trajectories against the intended trajectories according to the obtained control states. Although the traffic sampling can help reduce the monitoring overhead, existing approaches still require concurrently monitoring all network interfaces. In this paper, we propose a novel technique to improve the efficiency of the trajectory monitoring by only monitoring a subset of interfaces during each monitoring period. The proposed technique is generic and can be adopted by multiple trajectory error detection systems to improve their efficiency.

C. Router Group Monitoring

An important observation we made is that to detect a traffic trajectory error, it is sufficient to have just one monitored interface detect the error. In other words, monitoring all interfaces concurrently is unnecessary and overkill for trajectory error detection. Take Figure 1 as an example, if all interfaces are monitored, then routers R_4, R_5, and R_6 can all detect the same forwarding error, which creates unnecessary monitoring overhead.

This observation leads to the following question: Compared to the straight-forward setting of monitoring all interfaces concurrently, is it possible to detect the same trajectory errors in fewer sampling periods (i.e., faster) on average, without increasing the overall processing overhead, by monitoring fewer interfaces concurrently but each at a higher packet sampling rate? Conversely, is it possible to maintain the same detection speed, but reduce the overall processing overhead by monitoring fewer interfaces at the same packet sampling rate?

This paper studies these questions under a particular interface monitoring strategy we call *router group monitoring*. Suppose we model a network as a graph $G(V, E)$ where V is the set of vertices (routers) and E is the set of edges (links). A router group RG_i is a set of *connected* vertices such that $RG_i \subset V$. When monitoring a router group RG_i, interfaces on each cut edge $(u, v) \in E$ with $u \in RG_i$ and $v \in V \setminus RG_i$ are monitored. We informally refer to these interfaces as the periphery interfaces of a router group. The overhead of monitoring a router group thus depends on the number of periphery interfaces and the packet sampling rate used. The potential overhead saving comes from not monitoring those edges (u, v) with $u, v \in RG_i$.

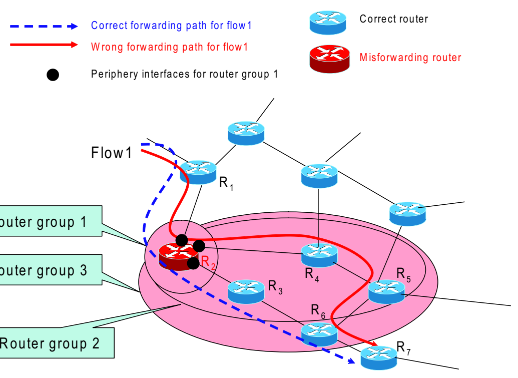

Correct forwarding path for flow1
Wrong forwarding path for flow1
Periphery interfaces for router group 1
Correct router
Misforwarding router
Flow1
R1
R2
R3
R4
R5
R6
R7
Router group 1
Router group 3
Router group 2

Fig. 1. Illustration of router group monitoring technique.

Figure 1 illustrates how the router group monitoring approach works. Router group 1 is a singleton router group. To monitor router group 1, we only need to monitor the three periphery interfaces represented by the black circles. By monitoring the router group 1, we can detect the forwarding error immediately because the flow 1 is leaving the group from a wrong periphery interface. When monitoring the router group 2, we can still detect this forwarding error. However, monitoring the router group 3 will not detect this specific error because it has been self-corrected inside the group. This tells us that a router group may not detect all errors originated from the inside of the group.

Multiple router groups can be concurrently monitored as long as the total processing overhead is below the desired ceiling. Note that concurrently monitored router groups RG_i and RG_j need not be disjoint. It is possible to choose a set of router groups that guarantee to detect all persistent trajectory errors. A sufficient condition is presented in Section IV. This result is intuitive because one can always choose N router groups where N is the number of routers and each group corresponds to a unique router in the network; this strategy simply degenerates into the monitoring of all network interfaces.

However, to determine whether router group monitoring improves efficiency, a number of questions must be addressed. First, how likely is a trajectory error inside a router group detectable at the periphery interfaces? Second, what factors affect the detection rate of a router group and how can we efficiently identify router groups that have high detection rates? Third, how can we choose a set of router groups that can guarantee error detection and achieve a low monitoring overhead? This paper systematically addresses each of these questions and show that router group monitoring has significant efficiency benefits for trajectory error detection.

D. Contributions and Findings

- We propose router group monitoring, which introduces a new spatial dimension to traffic trajectory error detection. That is, in addition to the dimension of varying the packet sampling rate to adjust the monitoring overhead, a new dimension to be considered is which network interfaces are to be monitored.
- To show that router group monitoring can be effective in practice, we analyze a large number of real network topologies by static computations and show that effective router groups with high trajectory error detection rates exist in all cases.
- We show that the router group size, the average router degree inside a group, and the number of exiting periphery interfaces are the key factors that influence a router group's detection rate. We develop an analytical model for quickly and accurately estimating the detection rates of different router groups. This model makes it possible to identify effective router groups efficiently.
- We propose an algorithm to select a set of router groups that can achieve guaranteed error detection and low monitoring overhead. We show that applying this algorithm to select router groups to be monitored can significantly improve the efficiency of trajectory error detection based on Trajectory Sampling or Fatih.

E. Road-map

The rest of this paper is organized as follows. In Section II, we study the effectiveness of router group monitoring in real topologies. In Section III, we derive an analytical model for predicting the effectiveness of a router group. In Section IV, we formulate the router group selection problem and present an efficient heuristic algorithm for router group selection. In Section V, we show the benefits of applying router group selection to Trajectory Sampling and Fatih. We discuss the related work in Section VI and conclude the paper in Section VII.

II. Effectiveness of Router Group Monitoring in Practice

A trajectory error represents a deviation from the intended network path and thus can potentially be detected at many interfaces in the network. Router group monitoring is a way to exploit this observation. Specifically, even if the trajectory of a packet starts to deviate from its intended path at a router inside a router group, the error may still be observable at the periphery interfaces of the router group. The effectiveness of the router group monitoring on detecting the three types of trajectory errors is discussed as follows:

- Dropping error - A dropping error simply drops all packets in the affected flow. Because a packet that is simply dropped in the middle of its trajectory will never leave the router group, by consistently observing packets missing from the intended exiting periphery interface, the error is easily detected. Thus, this paper will not focus on dropping errors.
- Filter-bypass error - A filter-bypass error causes a flow to bypass a packet filter that should drop it. When a filter-bypass error occurs inside a router group, whether it will be detected by monitoring the periphery interfaces depends on the distribution of packet filters inside the group. If the flow encounters another packet filter that is designed to drop it as well before it leaves the group, then the specific filter-bypass error will not be detected. On the other hand, if the flow leaves the group, then a periphery interface will see the unexpected flow so that the error will be detected. In practice, configuring the same packet filter on multiple routers along a path is not very common due to its inefficiency, so most filter-bypass errors will be easily detected. Thus, this paper will not focus on filter-bypass errors.
- Forwarding error - A forwarding error misforwards a flow to a wrong nexthop. A forwarding error can lead to two possible outcomes:
  1) Forwarding loop error: If a forwarding loop keeps a packet inside the router group, the packet will never leave the router group and can be detected just like a dropping error. If the forwarding loop takes the packet outside of the router group, if the exiting periphery interface is wrong, the error is detected. On the other hand, if the exiting periphery interface happens to be correct, the error is not detected by this router group.
  2) Detour error (no loop is formed): If the detour takes the packet outside of the router group via an incorrect exiting periphery interface, the error is detected. On the other hand, if the exiting periphery interface happens to be correct, the error is not detected by this router group. Therefore, a router group does not guarantee the detection of all forwarding errors that start inside the group. Different router groups can also have different error detection rates. Ultimately, multiple router groups must be chosen carefully to guarantee the detection of all trajectory errors and achieve low monitoring overhead. In this paper, we will focus on detecting forwarding errors because they are more subtle and more difficult to detect. Applying router group monitoring approach to detect other trajectory errors (e.g. filter-bypass error) is studied in detail in [30]. Our evaluation shows that the router group monitoring approach is also effective in detecting other types of trajectory errors.

A router at which an error occurs is called a *misbehaving router*. The misbehaving router's erroneous action such as dropping traffic, misforwarding traffic and allowing traffic to bypass filters is called a *trajectory error*. More formally, a misbehaving router is said to have one forwarding error with respect to a flow i denoted as F_i if it forwards all packets belonging to F_i to a wrong next hop interface. We perform a series of empirical experiments to understand the impact of router group monitoring on forwarding error detection.

A. Methodologies

1) Static Analysis Methodology: We first consider the case where only one forwarding error exists inside a router group.

Given a router group and a forwarding error inside the group, whether the forwarding error will be detected by monitoring the periphery interfaces of the router group can be decided using the following static analysis approach: starting from the misbehaving router, a hop-by-hop forwarding table lookup is used to decide the exiting interface where the mis-forwarded packet leaves the router group. If the exiting interface is the same as the original correct interface, then this error cannot be detected by using this router group. Otherwise, it can be detected because either the packet leaves from a wrong interface or a routing loop is formed. Similarly, if we want to know the overall effectiveness of one router group in detecting single forwarding error, we can calculate the *detection rate* of the router group as follows: for each router inside the group and for each possible destination in the network and for each possible wrong next hop interface for each destination, we introduce one forwarding error. Then a hop-by-hop forwarding table lookup is performed to decide whether the forwarding error can be detected. Thus, the detection rate can be calculated by dividing the number of detected errors by the number of total errors. Basically, given a network with N nodes and a router group with $|RG|$ nodes, $O(|RG| \times N \times (d-1))$ errors will be analyzed, where d is the average node degree and accordingly $d-1$ is the average number of wrong next hop interfaces. Because $|RG| = O(N)$ and $d = O(N)$, the complexity of exhaustively calculating the detection rate of one router group is $O(N^3)$ in the worse case.

Next, we consider the case where multiple forwarding errors exist in the router group. When two forwarding errors are independent from each other (i.e., they affect different flows), the detection rate for these errors is the same as in the single error case. On the other hand, if multiple forwarding errors do affect the same flow, we call them “dependent forwarding errors”. We only study the detection rate of multiple dependent forwarding errors. Given a network with N nodes and a router group with $|RG|$ nodes, in order to analyze K dependent forwarding errors ($K <= |RG|$), K distinct routers from the group will be selected, each of which will exhibit one forwarding error affecting the same flow. Each selected misbehaving router will mis-forward the flow to one wrong next hop interface. Similarly, a hop-by-hop forwarding table lookup is used to test whether the mis-forwarded packet can leave the router group from the original correct interface. The complexity of exhaustively analyzing all possible multiple forwarding errors is $O(C(|RG|, K) \times N \times (d-1)^K)$, where $C(|RG|, K) = |RG|!/K!(|RG| - K)!$. Suppose $K = 2$, then the worse case complexity is already $O(N^5)$.

For each network topology, we randomly choose router groups with different sizes and introduce forwarding errors as described above. We then can calculate an *average detection rate* for all the router groups. In Section II-B, for each topology, we calculate average detection rates for router groups with different sizes. For each router group size, we choose up to 500 random router groups in order to limit the computation time. We implement our analysis tool using Matlab scripts.

2) Topologies: To show the real-world detection performance of router group monitoring, we conduct forwarding error detection rate analysis using a large number of real network topologies, including Internet2, TEIN2 (Trans-Eurasia Information Network), iLight (Indiana’s Optical Network), GEANT (European research network), SUNET (Swedish University Network), Sprint North America backbone network composed of only Sprint global IP nodes, and six Rocketfuel [25] topologies. Table I summarizes the basic properties of each topology. For those topologies whose link weights are not available, we set all link weights as 1 to compute their routing tables.

Topologies:	# of nodes	# of edges	Degree	Link weight?
Internet2	9	13	(2, 2.9, 4)	Yes
TEIN2	11	11	(1, 2, 7)	No
iLight	19	21	(1, 2.2, 4)	No
GEANT	22	37	(2, 3.4, 9)	Yes
SUNET	25	28	(1, 2.2, 4)	No
Sprint (US)	28	46	(1, 3.2, 9)	No
RF-1	79	147	(1, 3.7, 12)	Yes
RF-2	87	161	(1, 3.7, 11)	Yes
RF-3	104	151	(1, 2.9, 18)	Yes
RF-4	138	372	(1, 5.4, 20)	Yes
RF-5	161	328	(1, 4.1, 29)	Yes
RF-6	315	972	(1, 6.2, 45)	Yes

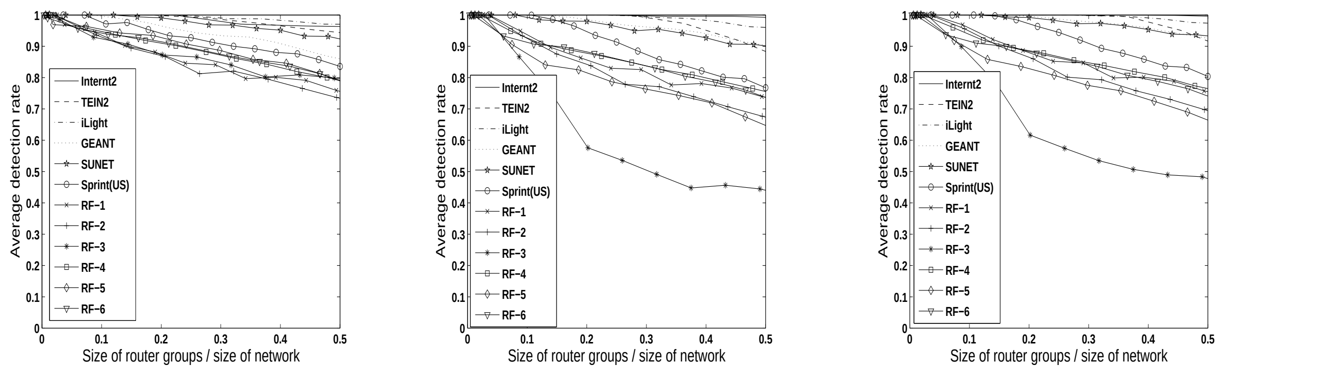
TABLE I
SUMMARY OF 12 REAL NETWORK TOPOLOGIES USED IN OUR EXPERIMENT. THE THREE NUMBERS IN THE DEGREE COLUMN ARE (MINIMUM DEGREE, AVERAGE DEGREE AND MAXIMUM DEGREE).

In addition to real network topologies, we have also conducted the same experiments on some representative synthetic topologies, such as power-law topologies (PLRG [6] and INET [27]), Hierarchical topologies (Transit-stub [29]) and random graphs. The results obtained from synthetic topologies are similar to those based on real network topologies shown in Section II-B. In this paper we will only present results based on real topologies. Results based on synthetic topologies are available in [30].

B. Detection Rate of Forwarding Errors

1) Single Forwarding Error: We first study how effectively the router group monitoring approach can detect single forwarding error in real network topologies. Figure 2(a) shows the results. We can make two observations from this graph. First, as the router group size increases, the fraction of detectable mis-forwarding cases decreases but only slowly. When the group size increases to 50% of the network size, the detection rates are still as high as 80% for most of the topologies. These results based on real topologies demonstrate that router group monitoring can be highly effective in practice.

2) Multiple Forwarding Errors: Next, we consider the case where multiple dependent forwarding errors exist in a router group. It is hard to predict the detection rate of multiple dependent forwarding errors because they can interact with each other. For example, after one router forwards a flow to a wrong path, the second misbehaving router on the wrong path might forward the flow back to the correct path. On the other hand, if the first misbehaving router fails to direct the flow to a wrong exiting interface, the second misbehaving router may increase the chance of the flow leaving from a wrong exiting interface by mis-forwarding it again. Therefore, the overall detection rate when having multiple dependent errors depends both on the network topology and the locations of the errors.



(a) One forwarding error (b) Two dependent forwarding errors (c) Three dependent forwarding errors

Fig. 2. Average forwarding error detection rates when applying router group monitoring on 12 real network topologies.

To better understand the detection rate for multiple dependent forwarding errors, we conduct the static analysis on the same set of real topologies. Specifically, we introduce 2 and 3 dependent forwarding errors on distinct routers inside each router group. The results are shown in Figure 2(b) and 2(c). We can see that some topologies have higher detection rates than the 1-error case, while the other topologies have lower detection rates. Results based on synthetic topologies also confirm that the detection rate of two dependent errors is not consistently better or worse than the one error case. However, it is worth noting that even multiple dependent errors co-exist and even if we use 50% of the nodes in the network as the router group, for most of the networks (except RF-3) we have studied, the average detection rate is still higher than 65%. Another interesting observation is that when the number of dependent errors increases from 2 to 3, the detection rates for all topologies also increase.

3) Relation Between 1-Error Detection Rate and Multi-Error Detection Rate: In this section, we ask the question: If a router group is effective in detecting one forwarding error, will it also be effective in detecting multiple forwarding errors?

To answer the above question, we first define the overlap ratio metric as the percentage of the 10% router groups with highest 1-error detection rates that also belong to the 10% router groups with the highest 2 or 3-error detection rates. Figure 3 then shows the result for all topologies. All topologies have high overlap ratios. Take the RF-6 topology as an example, the result shows that for the 10% router groups having highest 1-error detection rates, 89% of them are also among the 10% router groups having highest 2-error detection rates, and 88% of them are also among the 10% router groups having the highest 3-error detection rates.

In the rest of this paper, we use the 1-error detection rate to characterize the effectiveness of a router group.

III. Analytical Model for Router Group Effectiveness

As we have shown in Section II, router group monitoring can be highly effective in detecting trajectory errors in real topologies. However, for router group monitoring to be practical, those effective router groups with high trajectory error detection rates must be identified more efficiently than using the exhaustive hop-by-hop analysis approach.

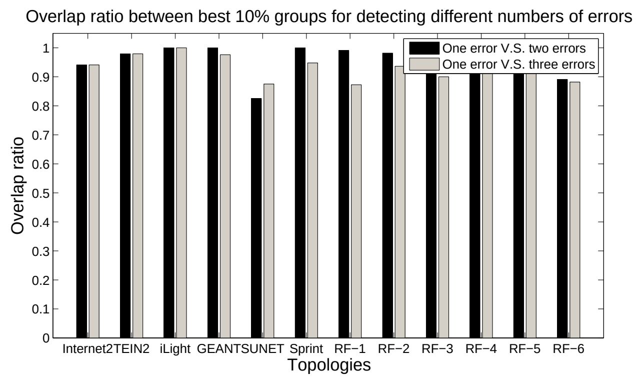


Fig. 3. Overlap ratio between the 10% router groups with highest 1-error detection rates and the 10% router groups with the highest 2 or 3-error detection rates.

One straight-forward way to avoid exhaustive analysis is to use sampling. For example, given a router group, instead of analyzing all errors, we only analyze a small subset of randomly selected errors to estimate the overall detection rate of the router group. Another approach is to develop an analytical model for quickly estimating the detection rates of different router groups. The analytical model should require much less computation than the static analysis approach. In this section, we will first present our analytical model, which only depends on some simple structural and routing metrics of router groups, and then we will compare the prediction accuracy of both the sampling approach and the analytic approach in Section III-B.

A. Contributing Factors of Trajectory Error Detection Rate

Three major contributing factors affecting the forwarding error detection rate have been identified as follows:

Router group size: As shown in Figure 2, the size of a router group is an important factor affecting its detection rate. Specifically, the average detection rate decreases with the increase of router group sizes. Given a router group, its size is easy to calculate. It is also not surprising that the size of a router group is important to its error detection rate. In a singleton router group with only one router, any error will be detected immediately. On the other hand, given a larger router group, a mis-forwarded packet is more likely to be self-corrected, i.e., it might fall back to its original routing path and leaves the router group from the original correct interface,

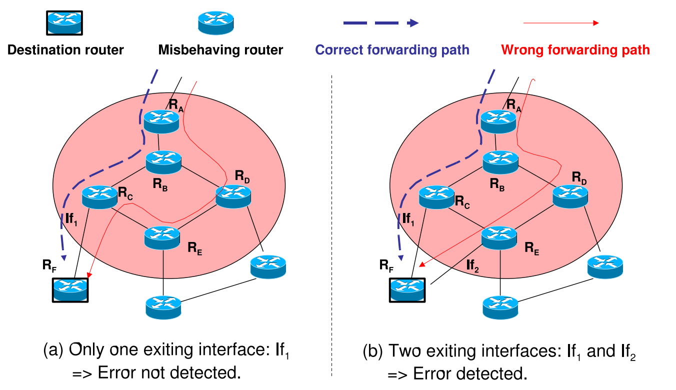

Destination router
Misbehaving router
Correct forwarding path
Wrong forwarding path
R_A
R_B
R_C
R_D
R_E
R_F
If_1
If_2
(a) Only one exiting interface: If_1 => Error not detected.
(b) Two exiting interfaces: If_1 and If_2 => Error detected.

Fig. 4. Illustration of how the number of exiting interfaces impacts the error detection rate.

thus the trajectory error might not be detected by this particular router group.

Number of exiting interfaces: Given a destination dst outside of the router group, a periphery interface If_i is called an *exiting interface* for dst if the interface's host router uses If_i as its direct next hop interface to route to dst. The router is called an exiting router accordingly. Given a particular destination, we can count how many periphery interfaces are exiting interfaces by scanning routing tables of routers having at least one periphery interface. The average number of exiting interfaces can be determined across all possible destinations. Intuitively, this factor characterizes how "diverse" the routing paths from inside the router group to a particular destination outside are. Please note that this metric is not the same as the number of periphery interfaces. One router group can have many periphery interfaces, but all the routers inside the group may only use a small number of periphery interfaces to route to any particular destination.

To illustrate why the number of exiting interfaces is important to a router group's error detection rate, Figure 4 (a) shows a router group with only one exiting interface If_1 with respect to the destination R_F. Since If_1 is the only exiting interface to R_F, when a forwarding error occurs (say R_B), it will be self-corrected by the router group (i.e., mis-forwarded packets end up leaving from the only exiting interface) unless a routing loop is formed. On the other hand, Figure 4 (b) shows a router group with two exiting interfaces (If_1 and If_2) for destination R_F, then a mis-forwarded packet is more likely to leave from the wrong exiting interface (If_2 in this example), allowing the error to be detected.

Connectivity of a router group: Given a router group, its connectivity is related to many topological characteristics of this group, such as the average node degree, the average outgoing degree (i.e., for each node, how many of its edges are connecting itself to nodes outside of the group), the average internal degree (i.e., for each node, how many of its edges are connecting itself to other nodes inside the group). All these metrics are very easy to calculate. Intuitively, the connectivity can impact how likely a mis-forwarded packet will be self-corrected inside the group and how likely a forwarding loop will be formed.

To illustrate why connectivity can impact the forwarding error detection rate, Figure 5(a) shows a router group with 5 routers, and the topology inside the router group has a cycle. There are two potential paths P_1 and P_2 between periphery interfaces If_1 and If_3. Assuming path P_1 is the correct path for a particular flow F, then flow F should enter the router group at the interface If_1 and leave at the interface If_3, following path P_1 inside the router group. However, if the router R_B has a forwarding error, it may forward the flow to router R_D as opposed to router R_B. The flow F will take path P_2 inside the router group, but it still leaves the group at interface If_3. In this case, this router group cannot detect router R_B's forwarding error. Generally, given a router group, if there is more than one path between an ingress interface and an egress interface, it is possible that some forwarding errors inside a particular router group cannot be detected from the periphery interfaces. Note that the same forwarding error may be detectable by using a different router group. In contrast, as shown in Figure 5(b), if a group of routers is connected in a tree topology, there is only one path between each ingress interface and egress interface. If the router R_B misforwards the flow F to the wrong path P_2, the flow F will either leave the router group at the wrong interface If_4, or be stuck between R_B and R_D (assuming R_B consistently misforwards the packet to R_D). Therefore, in a tree topology router group, any single forwarding error is guaranteed to be detected by monitoring the periphery interfaces because there are no redundant routing paths for a misforwarded packet to go back to the original correct path. Also, if a network has a full-mesh topology with all links having equal link weight, a forwarding error inside any router group is guaranteed to be detected. This is because all nodes have a one-hop path to any destination in the network, and so any forwarding error will result in the packet leaving the group from the wrong interface.

Generalizing these observations, intuitively, router groups in networks with tree-like topologies or full-mesh-like topologies will tend to have excellent error detection performance.

B. Analytical Model

Based on the three important factors identified above, we have developed an analytical model for accurately and quickly estimating the 1-error detection rate of a router group. We first define some notations we will use in our discussion. A router group's size is denoted as $|RG|$. The average number of exiting interfaces is e, which is also the number of exiting routers. The average outgoing degree and internal degree are d_{out} and

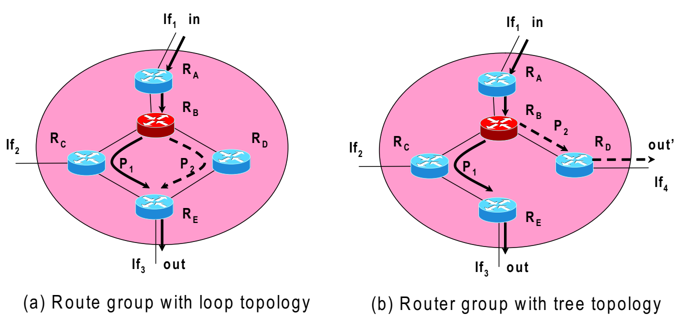


Fig. 5. The impact of router group connectivity on forwarding error detection.

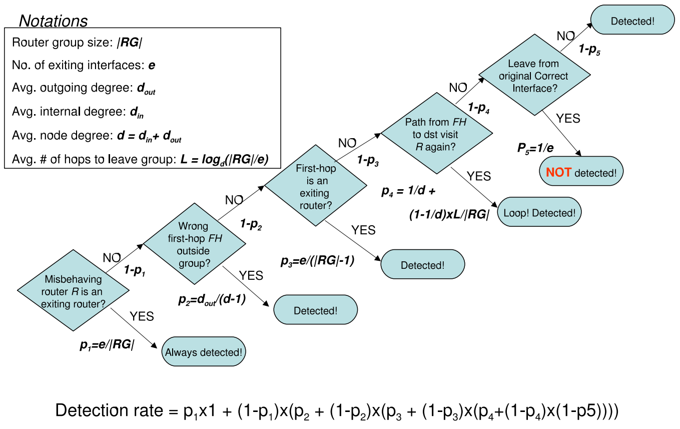

Notations
Router group size: |RG|
No. of exiting interfaces: e
Avg. outgoing degree: d_{out}
Avg. internal degree: d_{in}
Avg. node degree: $d = d_{in} + d_{out}$
Avg. # of hops to leave group: $L = log_d(|RG|/e)$
Misbehaving router R is an exiting router?
YES
$p_1=e/|RG|$
Always detected!
NO
$1-p_1$
Wrong first-hop FH outside group?
YES
$p_2=d_{out}/(d-1)$
Detected!
NO
$1-p_2$
First-hop is an exiting router?
YES
$p_3=e/(|RG|-1)$
Detected!
NO
$1-p_3$
Path from FH to dst visit R again?
YES
$p_4 = 1/d + (1-1/d)xL/|RG|$
Loop! Detected!
NO
$1-p_4$
Leave from original Correct Interface?
YES
$P_5=1/e$
NOT detected!
NO
$1-p_5$
Detected!
Detection rate = p_1x1 + (1-p_1)x(p_2 + (1-p_2)x(p_3 + (1-p_3)x(p_4+(1-p_4)x(1-p5))))

Fig. 6. Analytical formula for estimating error detection rate.

d_{in} respectively. The average node degree $d = d_{out} + d_{in}$. In deriving the model, we assume that each router has an equal chance to be the misbehaving one, and the misbehaving router will forward the affected flow to one random incorrect next-hop. This assumption is made to make sure that the errors analyzed in our model do not have a biased distribution. We also assume that any two routers inside a router group are equally likely to have a link connecting them. In addition, we assume that all links have equal weight and the correct trajectories follow shortest path routing. These assumptions are made to facilitate our model derivation. In a real network, these assumptions may not always accurately hold. However, our evaluation using 12 real network topologies in Section III-C shows that the derived model is robust and it can accurately estimate the detection rates of router groups even if those real topologies have different connectivity and non-uniform links weights.

We denote the misforwarding router as R_m. To accurately model the error detection rate of a router group, the first thing to note is that if R_m is an exiting router with respect to destination dst and is misforwarding packets destined to dst, then the error is guaranteed to be detected. Recall that an exiting router for dst is supposed to forward packets destined to dst directly out of the router group using its exiting interface. If it fails to forward the packet using its own exiting interface and assuming this is persistent, then the misforwarded packets will not leave the router group on the correct interface. Therefore, the forwarding error by an exiting router can always be detected. The probability that a router inside a router group is an exiting router is $p_1 = e/|RG|$.

However, if R_m is not an exiting router, its misforwarded packets may or may not leave the group from the correct exiting interface. When the non-exiting router R_m misforwards packets, it has the probability $p_2 = d_{out}/(d-1)$ to misforward packets directly out of the group using one of its outgoing edges, where $d-1$ is the number of all possible wrong next hops. In this case, the error will be caught by the device that is monitoring the corresponding periphery interface because the packets are observed from incorrect interfaces. On the other hand, R_m could misforward to a wrong next hop (also the first hop router) FH inside the router group. Since we assume only R_m in the router group is misbehaving, FH is a well-behaved router. Now we have two possibilities. The first possibility is that FH is an exiting router. The probability that FH is an exiting router is $p_3 = e/(|RG|-1)$, where $|RG|-1$ is the number of correct routers inside the router group. If this is the case, then FH will use its own exiting interface to route the packets out of the group. These packets therefore leave the group from an incorrect interface and will be caught because the correct trajectory follows the shortest path implies that FH does not lie on the correct trajectory. The other possibility is that FH is a non-exiting router. We model the length of the path $Path_{FH}$ from FH to its exiting router as $L = log_d(|RG|/e)$, where $|RG|/e$ is simply the average number of nodes using one particular exiting interfaces. If $Path_{FH}$ does not contain R_m, then the probability of $Path_{FH}$ leaves from the same exiting interface as R_m should have used is modeled as $p_5 = 1/e$, in which case the error cannot be detected by this router group. On the other hand, if $Path_{FH}$ does contain R_m, then a loop is formed, which will cause the error to be detected since the packet is missing from its expected exiting interface. We estimate the probability of $Path_{FH}$ containing R_m as $p_4 = 1/d + (1-1/d) \times L/|RG|$, where $1/d$ estimates the probability of FH sending the packet directly back to R_m forming a 1-hop loop and $L/|RG|$ estimates the probability of a path of length L contains a node R_m out of $|RG|$ possible nodes in total.

Figure 6 gives a summary of the model and the final analytical formula for estimating detection rates.

C. Prediction Accuracy of Using Model vs. Sampling

We now evaluate the accuracy of both model-based and sampling-based detection rate prediction as follows: Up to 100 router groups of each size are randomly chosen from each topology. We first use static analysis to calculate the exact detection rate for each chosen router group. Then we use our model to predict the detection rate for each router group and record the computation time required. For the sampling based approach, we sample different percentages (up to 50%) of errors and then predict the overall detection rate by analyzing only the sampled errors. We also record the computation time used for different sampling percentages. Figure 7 compares the average prediction errors (defined as the absolute difference between the predicted detection rate and the correct detection rate) of both approaches when they use the same amount of computation time. As can be seen, first of all, the model's average prediction error is smaller than 0.05 on most topologies. Therefore, the model successfully captures the important characteristics of the error detection. Second of all, given the same amount of computation time, the model can predict the detection rates more accurately for most topologies. On iLight and RF-2, the sampling based approach works only slightly better than the model based approach. Then for the ten topologies where our model works better than sampling, we study how much more time is needed to generate results as accurate as the model. Our results show that the sampling based approach generally needs a few times more computation time to have the same accuracy as the model-based approach.

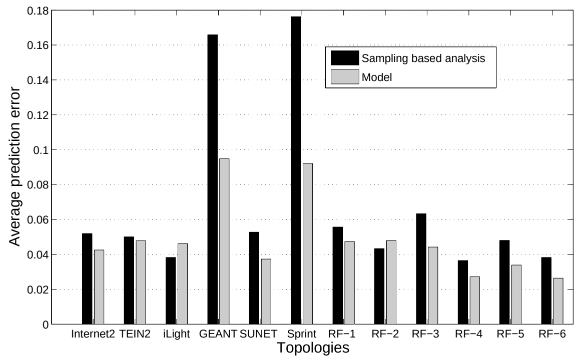


Fig. 7. Prediction errors of model-based and sampling-based approaches.

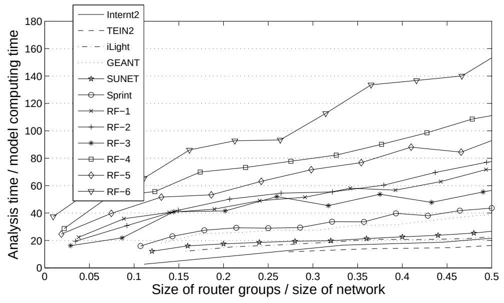


Fig. 8. Computational speedup of computing error detection rates using model versus computing error detection rates using static analysis approach.

For some topologies such as Sprint and GEANT, the sampling based approach needs 9 and 10 times more computation time to get the same prediction accuracy as the model.

The computation required by the model for computing the detection rate is significantly reduced compared against with the static analysis approach. Figure 8 shows the computation speedup comparison between the model based approach and the static analysis based approach. For all topologies except TEIN2, the speedup is over 20 times when a router group contains 50% of the nodes in the network. For some large router groups in the large topologies, the speedup is up to 153 times. For example, given a desktop computer with an Intel Pentium 4 3.0 GHz CPU, 9 hours of computation time is used to compute the detection rates of 1000 random router groups in RF-6 topology by analyzing all errors inside each group, while it only costs 5 minutes of computation time for our analytical model. As expected, the computation saving of using our model increases when the network becomes larger.

Another useful property of our model is that the pair-wise ranking order among router groups is mostly preserved, which is very important to our router group selection algorithm in Section IV, where we use predicted detection rates to help select the most effective router groups. Specifically, for each pair of router groups, we will predict which one has a larger detection rate using our model and then validate the results using the detection rate calculated by the static analysis approach. Figure 9 shows the percentage of router group pairs whose order is preserved by the model. For example, the model correctly predicts the ranking order for 89.2% of router group pairs in the Sprint topology.

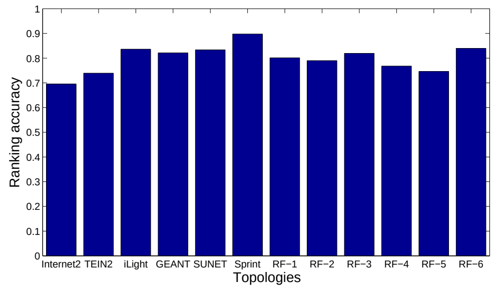


Fig. 9. The model accurately preserves the ranking order among pairs of router groups.

IV. Router Group Selection Algorithm

We have demonstrated that router group monitoring is effective in detecting traffic trajectory errors. We have also proposed a model to predict the detection rates of router groups. The next problem is to design an algorithm to choose a suitable set of router groups for the system to monitor for each monitoring period.

As explained in Section I, existing traffic trajectory monitoring algorithms monitor different subsets of packets during different monitoring periods. If during each monitoring period, x% of packets are monitored, then $\frac{100}{x}$ monitoring periods are needed to cover all traffic. Similarly, router group monitoring can also be performed period by period. However, in order to reduce monitoring overhead, in router group monitoring, only up to M interfaces are monitored during each monitoring period, where M is no larger than the total number of interfaces of the network and the M interfaces are periphery interfaces of the set of monitored router groups.

A good router group selection algorithm should (1) provide complete trajectory error detection coverage, which is the *correctness* requirement elaborated in Section IV-A, and (2) detect errors as quickly as possible, which is essentially the *optimality* requirement discussed in Section IV-B.

A. Correctness of Router Group Selection Algorithm

As we explained, when using router group monitoring, some interfaces (M) are monitored during each monitoring period. Thus, the first concern of the router group selection algorithm is whether it can guarantee complete trajectory error detection coverage. One straight-forward way to satisfy this requirement is to treat each single router as a router group and then always include all the singleton router groups for monitoring. This is however unnecessary. We give a more general sufficient condition as follows.

Lemma 1: To guarantee that all observable trajectory errors[1] are eventually detected, it is sufficient to select a set of router groups such that every router interface f_{ij} on a node v_i connecting to a node v_j is an end of a cut edge $(v_i, v_j) \in E$ of a selected router group RG, with $v_i \in RG$ and $v_j \in V \setminus RG$.

[1]A fundamental requirement for trajectory error detection in any approach is that the evidence of an trajectory error must be observable by other monitoring nodes. This paper does not address errors that are not observable. For example, if one router mistakenly drops a packet destined to itself, then this error cannot be detected because it is not observable from outside.

Intuitively, f_{ij} is a periphery interface of a router group RG facing outward.

Every error, whether it is a mis-forwarding, or a packet dropping or a filter-bypass exhibits itself in one of two ways: (1) a packet that should have been observed on an interface is not observed; (2) a packet that should not have been observed on an interface is observed. If a router interface is a periphery interface of a router group RG facing outward, then that interface's behavior is monitored when router group RG is monitored. Therefore, any error involving that interface will be caught.

B. Complexity of Optimal Router Group Selection

Given the sufficient condition, we can now easily tell whether a set of router groups can provide complete error coverage. The next question is: how should we select router groups to monitor during each monitoring period so that we can not only achieve complete error coverage but also iteratively monitor the *smallest* number of monitoring periods? This is the *optimality* requirement of the router group selection problem.

Minimizing the total number of monitoring periods while providing complete error coverage is a hard problem. The reason can be intuitively explained as follows. Suppose each interface f_{ij} is involved in some number of errors. When the interface f_{ij} is inside (i.e. not on the periphery of) a router group RG_k, those errors involving f_{ij} can be detected with some probability $w(RG_k, f_{ij}) \in [0\ 1]$. Therefore, once RG_k is selected for monitoring, the usefulness of monitoring other router groups that also contain f_{ij} will decrease accordingly. This interdependence makes it hard to determine an optimal selection of router groups.

1) Definition of the monitoring problem: We now formalize the router group selection problem as follows.

Notations:

- Let $G = (V, E)$ be a graph, where V is a set of nodes
$$V = \{v_1, v_2, \cdots, v_n\}$$
and E is a set of edges $E \subseteq V \times V$.
- Each node v_i is associated with a set of *interfaces*
$$F_{v_i} = \{f_{ij}\ :\ (v_i, v_j) \in E\}$$
The set of all the interfaces in the graph $G = (V, E)$ is $F = \bigcup_{v_i \in V} F_{v_i}$ [2]. For any subset $A \subseteq V$,
$$F_A = \{f_{ij}\ :\ (v_i, v_j) \in E \wedge (v_i \in A) \wedge (v_j \notin A)\}$$
- Given function α, for any subset $A \subseteq V$ the monitoring weight function is defined as[3]: for each $f_{ij} \in F$,
$$w(A, f_{ij}) = \begin{cases} 1 & \text{if } f_{ij} \in F_A \\ \alpha(A, i, j) \in [0,1] & \text{if } v_i \in A \wedge v_j \in A \\ 0 & \text{if } v_i \notin A \wedge v_j \notin A \end{cases}$$
- A *Monitoring* is defined as a multiset of sets of subsets of V[4]:
$$\begin{aligned} \mathcal{A} &= \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m\} \\ &= \{\{A_{11}, \ldots, A_{1n_1}\}, \{A_{21}, \cdots, A_{2n_2}\}, \ldots, \\ &\quad \{A_{m1}, \cdots, A_{mn_m}\}\} \qquad (A_{ij} \subseteq V) \end{aligned}$$
where $m \geq 1$ and $n_i \geq 1$.

Optimization objective:

Find the smallest *Monitoring* $\mathcal{A}$ such that for any $\mathcal{A}'$ we have $|\mathcal{A}| \leq |\mathcal{A}'|$, where $\mathcal{A}$ and $\mathcal{A}'$ satisfy the following two constraints:

- Given a *Monitoring* $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m\}$, for each $1 \leq i \leq m$, $\mathcal{A}_i \in \mathcal{A}$,
$$|\bigcup_{A \in \mathcal{A}_i} F_A| \leq M$$
Where $M \geq 1$.
- Given a *Monitoring* $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m\}$ and a constant $0 \leq \beta \leq 1$, for each $f_{ij} \in F$ $(1 \leq i, j \leq |V|)$, [5]:
$$1 - \prod_{1 \leq k \leq m} \prod_{A \in \mathcal{A}_k} (1 - w(A, f_{ij})) \geq \beta$$

We generally want to find the smallest *Monitoring* given $\beta = 1$ and a small constant M.

In the above problem formulation, during monitoring period i, a set of router groups $A_i = A_{i1}, A_{i2}, ..., A_{in_i}$ are monitored concurrently, where A_{ij} and A_{ik} could overlap with each other. We have studied a special case of the above problem, where for any $j, k \in [1\ n_i]$ and $j \neq k$, A_{ij} and A_{ij} always have no overlap. We have proved that as long as $M < |F|$, the above special case is a NP hard optimization problem [18]. Please refer to [30] for the complete proof. We believe the general case is also NP hard and we are currently working on the proof.

C. Heuristic Algorithm for Router Group Selection

Given the above definition, we now give a heuristic algorithm for selecting a set of router groups that achieves complete coverage, has bounded concurrent monitoring overhead, and in practice provides timely error detection.

Input of the algorithm: A positive integer M, a set of n router group candidates denoted as $RG_{candidates} = RG_1, RG_2, ..., RG_n$, and the $w(RG_k, f_{ij})$ function defined in Section IV-B1.

M is the maximum number of interfaces that the system can concurrently monitor and it should be determined by the operator based on resource constraints. We assume that the maximum degree of any router in the network is no larger than M. $RG_{candidates}$ should contain a large number of diverse router groups of different sizes in order to provide enough opportunity for the selection algorithm to explore. We cannot include all possible router groups into $RG_{candidates}$ for large

[2]The set of interfaces is decided by the set E

[3]Intuitively, the weight $w(A, f_{ij})$ means that the errors involved with the interface f_{ij} can be detected using group A with probability of $w(A, f_{ij})$. Trajectory errors on periphery interfaces can always be immediately detected, while errors on other interfaces may or may not be detected.

[4](1) Nodes in A_{ij} may not necessarily be connected. (2) A_{ij} and A_{ik} may have overlapped nodes.

[5]Intuitively, this means that after m groups of monitoring, all the error involved on the interface f_{ij} have been detected with a probability of at least β.

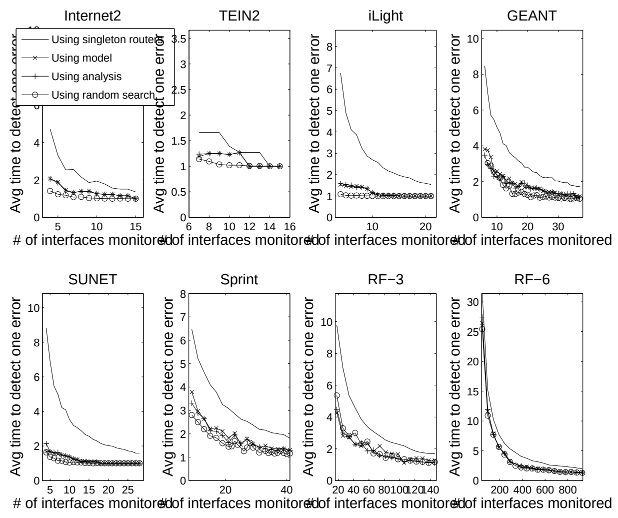


Fig. 10. Comparison of average error detection speeds of different router group selection approaches.

networks. Thus, we first randomly generate a number of router groups of each size, and then select up to K router groups from each size with the highest predicted detection rates. All singleton routers are always included in $RG_{candidates}$, which is important for guaranteeing that the selection algorithm eventually terminates. The $w(RG_k, f_{ij})$ function specifies that if RG_k is monitored, then the errors involving interface f_{ij} can be detected with a probability of $w(RG_k, f_{ij})$.

Output of the algorithm: Given M, function $w(\cdot)$ and $RG_{candidates}$, the output of the algorithm should be m sets of router groups, $T_1, T_2, .., T_m$, where $T_i \subseteq RG_{candidates}$. Then we can iteratively monitor all m sets of router groups one by one. If we can sample x% packets at each moment, then $\frac{100}{x}$ periods are needed for each T_i. That is, in total, $m \times \frac{100}{x}$ monitoring periods are needed to cover all traffic.

Algorithm's intuition: The main idea of our heuristic algorithm is to keep greedily selecting a set of router groups that have the potential to detect most uncovered errors to form a new set of router groups T_i until the sufficient condition is satisfied. We define an uncovered error function $E(f_{ij}) \in$[0 1] on each interface f_{ij} to represent the fraction of uncovered errors on f_{ij} at the current moment. At the beginning of the algorithm, none of interface f_{ij}'s errors have been covered by any selected router group, so $E(f_{ij}) = 1$. Once a router group RG_k containing $f_{i,j}$ has been selected for monitoring, we update $E(f_{ij})$ as follows: $E(f_{ij}) = E(f_{ij}) \times (1 - w(RG_k, f_{ij}))$.

Suppose $RG_{selected}$ is the set of selected router groups at this moment. Now we can define the selection weight of a router group RG_k as follows:

$$W(RG_k) = \begin{cases} 0 & if\ RG_k \in RG_{selected} \\ \sum\limits_{f_{ij}} w(RG_k, f_{ij}) \times E(f_{ij}) & if\ RG_k \notin RG_{selected} \end{cases}$$

The router group selection algorithm is as follows:

```
01: $RG_{candidates} = \{$All singleton routers$\}$;
02: FOR $m = 2 : |V| - 1$
03:     Randomly generate up to $T$ router groups containing
        $m$ routers and $\leq M$ periphery interfaces
        and then select up to $K \leq T$ router groups with highest
        predicted detection rates and put into $RG^m_{candidates}$;
04:     $RG_{candidates} = RG_{candidates} \cup RG^m_{candidates}$;
05: END-FOR
06: $RG_{selected} = \{\ \}$;
07: $A = \{\ \}$;
08: $E(f_{ij}) = 1$, for $\forall f_{ij} \in F$;
09: $period = 1$;
10: WHILE($\sum E(f_{ij}) > 0$)
11:     AvailableIFs = $M$;
12:     $A_{period} = \{\ \}$;
13:     WHILE AvailableIFs>0 AND $\sum E(f_{ij}) > 0$
14:         Find $RG_i \in RG_{candidates}$ with largest $W(RG_i)$
            and with $\leq$ AvailableIFs periphery interfaces,
            if multiple choices exist, pick the largest group,
            if no such choice exits, break the WHILE loop;
15:         $RG_{candidates} = RG_{candidates} \backslash \{RG_i\}$;
16:         $RG_{selected} = RG_{selected} \cup \{RG_i\}$;
17:         $A_{period} = A_{period} \cup \{RG_i\}$;
18:         $\forall f_{ij}, E(f_{ij}) = (1 - w(RG_i, f_{ij})) \times E(f_{ij})$;
19:         Update $W(RG_j)$ for $\forall RG_j \in RG_{candidates}$;
20:         AvailableIFs -= # periphery interfaces of $RG_i$;
21:     END-WHILE
22:     $A = A \cup A_{period}$;
23:     $period = period + 1$;
24: END-WHILE
25: RETURN $A$;
```

Algorithm termination and correctness: Since we assume M is no smaller than the largest router degree in the network, each router is eligible to form a singleton router group while not violating the resource constraint. Since $RG_{candidates}$ includes all singleton router groups, the selection algorithm can always return the singleton router groups. Therefore, the algorithm is guaranteed to terminate and return a set of router groups that has complete coverage.

D. Performance of Heuristic Router Group Selection Algorithm

In this section, we evaluate the performance of our heuristic algorithm. In the experiments, we use $K = 10$ to initialize $RG_{candidates}$. We study the performance of the algorithm using various M, as long as M is no smaller than the maximum degree of the network topology.

We use two different approaches of estimating function $w(RG_k, f_{ij})$. The first approach is based on static analysis, so we can accurately know the $\alpha(RG_k, i, j)$ function. The second approach is based on our detection rate model. Given a router group RG_k, suppose its predicted detection rate is $detection_k \in$ [0 1] and suppose the router group RG_k contains $p_k \in$ [0 1] fraction of periphery interfaces and accordingly $(1 - p_k)$ faction of non-periphery interfaces. If we assume all internal non-periphery interfaces in RG_k have the same $\alpha(k)$ values, then we have $detection_k = p_k \times 1 + (1 - p_k) \times \alpha(k)$, i.e., $\alpha(k) = (detection_k - p_k)/(1 - p_k)$.

As a baseline for comparison, we also include the performance of singleton router based selection algorithm, whose $RG_{candidates}$ only contains all singleton router groups. In order to estimate how close our heuristic algorithm is to the real optimal group selection, we also compare with a bounded random search based approach. Specifically, given a topology and its $RG_{candidates}$, we will randomly select a multiset of sets of router groups for monitoring and then we can compute a corresponding average detection speed by introducing $10,000$ random forwarding errors uniformly distributed across all nodes for all possible destinations. We repeat this random group selection process $10,000$ times and keep the best detection speed we found. Please note that performing $10,000$ random search is very expensive. For example, given the RF-6 topology, 66 hours of computation time is used to finish on a desktop computer with an Intel Pentium 4 3.0 GHz CPU. On the other hand, it only costs 16 minutes of computation time for our algorithm.

For each topology, we introduce up to $10,000$ forwarding errors uniformly distributed across all routers for all possible destinations one by one. We then statically analyze to see how many monitoring periods it will take for each approach to detect each introduced error. We then present the average detection speed of all four approaches on different topologies in Figure 10. We omit the results for RF-1, RF-2, RF-5 as they are qualitatively similar to those of RF-3. The results for RF-4 is qualitatively similar to those of RF-6 and are also omitted. As we can observe, first of all, the detection speeds of the approach based on the model predicted detection rates are very close to the one using static analysis across all topologies. Secondly, the detection speeds of our approach are also very close to the bounded random search based approach, though our approach requires much less computation time. For some topologies such as RF-3, our heuristic algorithm is better for some M. This indicates our heuristic algorithm is effective in quickly selecting a good set of router groups for monitoring. Thirdly, our algorithm outperforms the singleton router groups based approach for all topologies. Especially when M is a small value, our approach can detect an error a few times faster than the singleton router group based approach. This performance gain comes from the fact that we are covering much more routers at any moment, though both approaches monitor the same number of interfaces and have the same overhead.

E. Discussion

In our problem formulation, we assume no constraints on which routers can be used for monitoring, that is, all routers are assumed to be homogeneously powerful. However, routers in real networks might be very heterogeneous. For example, some routers may even not have the monitoring capability. Fortunately, we can always use standalone passive traffic monitoring devices (e.g., [2]) to tap on the corresponding network links to perform the monitoring function. In addition, some low-end routers might only afford up to a certain sampling rate due to resource constraints in hardware or software. In this case, we can either use standalone passive traffic monitoring devices for monitoring or we need to carefully set the sampling rate on all periphery interfaces to not to exceed the the required resource constraints on the slowest monitoring device. If certain routers need to be taken offline for the scheduled maintenance, then the operator should plan ahead and calculate a new set of router groups for monitoring according to the specific topology change. If topology change is caused by other dynamic network events such as link failures, it can be learned from the dynamic routing protocol messages such as OSPF LSAs. To quickly respond to the dynamic topology change, different sets of router groups with respect to different potential network events should be computed in advance as well. How to incrementally update the set of monitored router group to efficiently accommodate unexpected dynamic events is one of our future work.

V. Applications of Router Group Monitoring

In this section, we show how the router group monitoring technique can improve the efficiency of trajectory error detection based on Trajectory Sampling and Fatih. The basic Trajectory Sampling algorithm monitors all interfaces in the network and samples the same subset of packets at the same time. Then, information about sampled packets is sent to a centralized collector for analysis. The basic Fatih algorithm, on the other hand, monitors all interfaces that are used in forwarding packets, although as we shall see this is nearly the same as monitoring all interfaces in practice. Fatih also samples the same subset of packets at the same time. The fingerprints of the sampled traffic belonging to each network path will be exchanged among the monitors along that path for analysis.

The router group monitoring technique can be used to select a subset of network interfaces to be monitored under Trajectory Sampling or Fatih. This translates into reduced monitoring overhead and/or faster trajectory error detection without sacrificing the completeness of coverage.

A. Applying to Trajectory Sampling

In Trajectory Sampling, all network interfaces in the network will sample the same subset of packets (say, 1% of all traffic) during the same monitoring period. Different subsets of packets will be sampled for different monitoring periods to achieve complete coverage.

1) Scenario One: Improve Detection Speed While Keeping the Reporting Traffic Overhead Constant: In this scenario, we want to keep the reporting overhead (i.e., how many messages are sent to the collector per period) constant so that we do not overwhelm the collector.

Suppose we can vary the sampling rate in a small range from 1% to 5%. Can router group monitoring improve the trajectory error detection speed while keeping the reporting overhead constant? To maintain the same reporting overhead, when we increase the sampling rate m times, we decrease the number of concurrently monitored interfaces by m times accordingly. The overall reporting overhead is maintained at the same level as sampling all interfaces in the network with a 1% rate.

Figure 11 shows the result. If we use a 5% sampling rate and allow the concurrent monitoring of 20% of the interfaces, the detection speedup over baseline Trajectory Sampling (i.e., sampling 1% on all interfaces) is at least 2 times and for some topologies the detection speedups are more than 4 times. The detection speedup comes from the fact that when we increase the sampling rate, we can rotate the set of monitored router groups more quickly. For example, if we use a 5% sampling

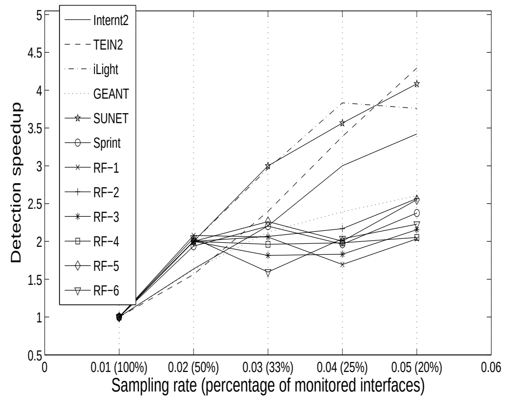


Fig. 11. Detection speedup when varying the sampling rate and the maximum number of interfaces concurrently monitored.

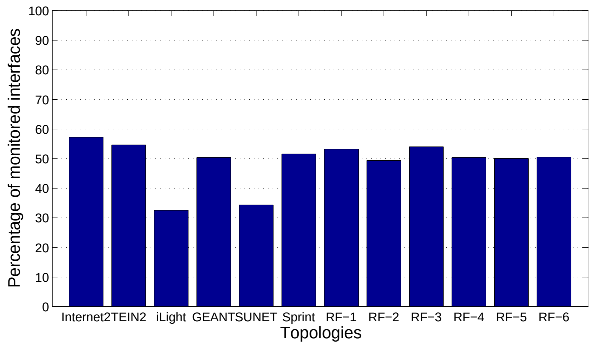


Fig. 12. Percentage of monitored interfaces required to achieve the same detection speed as the original Trajectory Sampling.

rate, we only need to monitor each set of router groups 20 periods then we can rotate to a new set of router groups that can detect another set of errors. Specifically, taking SUNET as an example. If we assume that each monitoring period lasts one minute and the router group monitoring approach monitors 20% of all the interfaces with a sampling rate of 5%, then it will take the router group monitoring approach 25 minutes to detect all errors, while it will take 105 minutes for the original Trajectory Sampling.

2) Scenario Two: Reduce Fraction of Monitored Interfaces While Keeping Same Detection Speed: In this scenario, we assume a fixed 1% sampling rate. Then we want to study what fraction of interfaces we have to monitor to keep the same detection speed as monitoring all interfaces simultaneously. Figure 12 shows the result. As can be seen, for certain topologies such as iLight, concurrent monitoring of 33% of the interfaces are enough to provide the same detection speed as baseline Trajectory Sampling. For most of the topologies, monitoring roughly 50% of the interfaces concurrently is enough to detect errors as quickly as baseline Trajectory Sampling. Specifically, taking RF-6 as an example. Assuming that each interface forwards 13,000 active flows per second on average [5]. Given a 5% sampling rate, each interface can sample 650 active flows per second on average. Because each NetFlow record is 64 bytes, each interface will generate 332.8 Kbps of traffic. Since there are a total of 1944 interfaces in RF-6 topology, 646.9 Mbps of reporting traffic will be generated. On the other hand, the router group monitoring approach will only generate about 329.9 Mbps of reporting traffic while having the same detection speed.

B. Applying to Fatih

In Fatih, each router r_i needs to maintain certain traffic information for *each* 3-path-segment containing itself. A 3-path-segment is a subpath with length 3. The traffic information Fatih maintains for each path segment is the fingerprints (e.g., hash values of the packets) of all the packets r_i forwarded along the monitored path segment. Periodically, router r_i exchanges the fingerprints information with other routers on the same 3-path-segment. Because all 3-path segments r_i, r_j, r_k are monitored, then if r_j dropped or misforwarded packets, r_i and r_k can detect this error when they exchange traffic information.

For the purpose of trajectory error detection, we can use the router group monitoring technique to reduce the monitoring overhead by only having each periphery router to maintain information about what traffic it will forward to other periphery routers in the same router group. Therefore, while in baseline Fatih each router keeps a set of information for each path segment, in contrast, with router group monitoring, only periphery routers need to maintain information for other periphery routers.

We evaluate the benefits of applying router group monitoring to Fatih. First of all, we compare the number of interfaces monitored with and without router group monitoring while keeping the detection speed the same in Figure 13. Since Fatih needs to monitor every 3-path segments, for many topologies, all interfaces need to be monitored. The interfaces will not be monitored if the corresponding link is not used or only used in an end-to-end path with length 2. Applying router group monitoring allows much fewer interfaces to be monitored while having the same detection speed and detection accuracy.

Next we evaluate the fingerprints communication overhead saving after using router group monitoring. In this experiment, we assume the same amount of traffic is sent between each pair of nodes in the network, and we study the fingerprint exchanging overhead with and without router group monitoring. The results are shown in Figure 14. The communication overhead of the baseline Fatih is normalized to 100 units. As can be observed, applying router group monitoring can reduce dramatically the fingerprint communication overhead for all topologies. For certain topologies, the overhead reduction is more than 80%. To understand the absolute reporting overhead reduction, we first take RF-6 as an example. Following the same assumption in Section V-A2, we assume that ten packets on average will be sampled for each flow. If each hash value is 8 bytes, then a total of 808 Mbps of traffic will be sent to the collector by all links. By employing router group monitoring approach, the reporting overhead can be reduced from 808 Mbps to 266 Mbps while keeping the same detection speed.

VI. Related Work

Network measurement and monitoring are important for many network management applications. However, measurement and monitoring often incur high overhead. Therefore, a

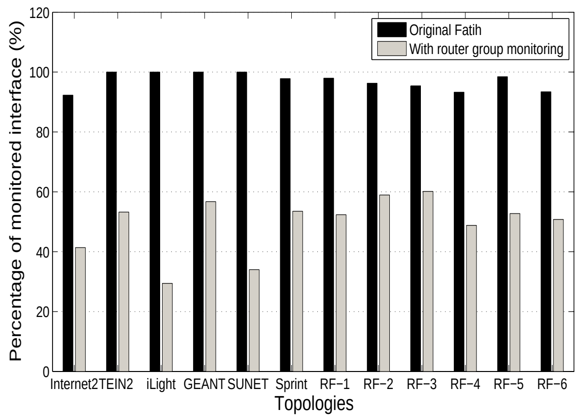


Fig. 13. Router group monitoring helps Fatih reduce monitored interfaces.

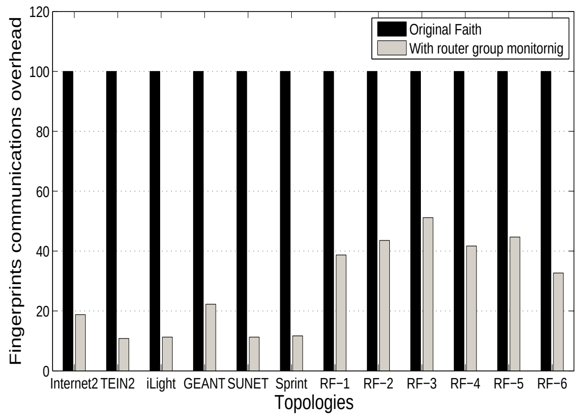


Fig. 14. Router group monitoring helps Fatih reduce communication overhead.

constant theme in many related research is to improve the efficiency of measurement and monitoring techniques. The goal of our paper is to specifically improve the efficiency of traffic trajectory monitoring. In the following, we discuss some previous work on improving the efficiency of monitoring and measurement for other important applications.

WATCHERS [7], [15] maintains several packet counters at routers and uses inconsistencies found in these counters among different routers to detect forwarding errors. Because it only uses course-grained counters, it is only capable of detecting dropping errors. Sekar *et al* [23] presented a new flow monitoring system, cSamp. cSamp can improve the flow monitoring coverage by enabling routers to coordinate and to sample different flows. Their goal is however not to identify the trajectory errors in flows. Therefore, cSamp can only tell which flows are in the network, but it does not know the actual trajectories of the monitored flows. Lee *et al* [19] presented a secure split assignment trajectory sampling (SATS) technique. The idea is to enhance trajectory sampling by letting each pair of routers to sample different subsets of packets to improve monitoring coverage. However, SATS cannot detect forwarding error unless a forwarding loop is formed causing some packet loss. In addition, it only detects packet dropping error with a certain probability. On the other hand, our approach can detect both forwarding and dropping errors. Like our router group monitoring technique, cSamp and SATS also introduce a spatial dimension to their solution in the sense that different parts of the network perform different heterogeneous tasks to improve the overall efficiency of traffic monitoring. At the interface traffic sampling level, Estan *et al* [11] proposed a set of efficient techniques to adapt the NetFlow sampling rate in order to better control resources consumption. Kompella and Estan [17] proposed an efficient flow measurement solution called Flow Slices to control and reduce CPU usage, memory usage and reporting bandwidth of flow measurements. Our router group monitoring technique currently assumes packet level monitoring. Applying these flow based monitoring techniques can potentially improve the efficiency of trajectory error detection further. These efficient interface-level sampling techniques are orthogonal and complementary to our work.

Router group monitoring could also be viewed as a sort of traffic monitor placement technique because it identifies interfaces that need or need not be monitored for trajectory error detection. However, our technique is only designed for the trajectory error detection problem. The selected interfaces always form a boundary around a group of routers, so we can track how the traffic flows through the network. The more general monitor placement problem has been extensively studied for various problem settings. However, all these monitor placement techniques aim to sample more flows, instead of learning the actual spatial trajectories of flows. That is, they are designed to sample different flows at different locations while our approach is designed to sample the same flow at different locations so that we can infer their complete trajectories. Horton and Lopez-Ortiz [14] addressed the monitor placement problem in an active monitoring infrastructure to efficiently measure delays and detect link failures. Suh *et al* [26] used optimization techniques to place monitors and set the sampling rates in order to maximize the fraction of IP flows being monitored. They first find the links that should be monitored and then run another optimization algorithm to set sampling rates. Chaudet *et al* [9] not only studied the tap device placement problem for passive monitoring but also the beacon placement problem for active monitoring. Their goal is to minimize the number of tap devices used for passive monitoring and to find the optimal locations for placing the beacons. Similarly, Cantieni *et al* [8] proposed mechanisms to optimally select links to be monitored and select sampling rates in order to achieve specific measurement tasks with high accuracy and low overhead. Jackson *et al* [16] studied the monitor placement problem using the current Internet topology. Their goal is to choose a set of locations to maximize the chance of covering all possible communication pairs in the Internet. Note that in general, how to optimally choose interfaces to monitor for trajectory error detection is still an open problem. Zang *et al* [28] investigates the problem of deploying NetFlow with optimized coverage and cost in an IP network. It aims to solve the Optimal NetFlow Location Problem (ONLP) for a given coverage ratio. However, it only samples flows at fixed points instead of monitoring their actual spatial trajectories.

VII. Conclusion

This work started with a simple observation: To detect a traffic trajectory error in a network, it is unnecessary to monitor all network interfaces. However, how to exploit this observation was not entirely obvious. This paper has explored one class of strategy called router group monitoring. To understand the potential of this strategy, we have studied numerous real network topologies and found that router group monitoring is surprisingly effective. To make this idea practical, we have derived an analytical model to predict the effectiveness of a router group as well as designed an efficient algorithm for selecting sets of router groups with complete error coverage and fast error detection under monitoring resource constraints. The analytical model provides key insights on the factors that determine the error detection rate. Our router group selection algorithm, when applied to Trajectory Sampling, can improve detection speed by up to a factor of 4, and when applied to Fatih, can reduce the communication overhead by up to 85%. Interestingly, router group monitoring is just one of possibly many interface selection strategies that remain to be explored.

References

[1] Cisco Security Advisories and Notices. http://www.cisco.com/en/US/products/products_security_advisories_listin%g.html.

[2] FlowMon: Comprehensive Solution for NetFlow Monitoring. http://www.invea-tech.com/products/flowmon.

[3] Quagga Bugzilla. http://bugzilla.quagga.net/.

[4] Quagga Software Routing Suite. http://www.quagga.net/.

[5] Sprint IP Data Analysis Research Project. https://research.sprintlabs.com/packstat/packetoverview.php.

[6] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In Proccedings of ACM Symposium on Theory of Computing, pages 171–180, 2000.

[7] K. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. Olsson. Detecting Disruptive Routers: A Distributed Network Monitoring Approach. In IEEE Network, 1998.

[8] G. Cantieni, G. Iannaccone, C. Barakat, C. Diot, and P. Thiran. Reformulating the monitor placement problem: optimal network-wide sampling. In ACM CoNEXT, December 2006.

[9] C. Chaudet, E. Fleury, and H. Rivano. Optimal Positioning of Active and Passive Monitoring Devices. In ACM CoNEXT, October 2005.

[10] N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. In Proc. ACM SIGCOMM, pages 271–282, 2000.

[11] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a Better NetFlow. In ACM SIGCOMM 2004, August 2004.

[12] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. NSDI, 2005.

[13] A. Feldmann and J. Rexford. IP Network Configuration for Intradomain Traffic Engineering. IEEE Network, 2001.

[14] J. Horton and A. Lopez-Ortiz. On the number of distributed measurement points for network tomography. In USENIX IMC, November 2003.

[15] J. Hughes, T. Aura, and M. Bishop. Using conservation of flow as a security mechanism in network protocols. In IEEE Symposium on Security and Privacy, May 2000.

[16] A. Jackson, W. Milliken, C. A. Santivanez, M. Condell, and W. Strayer. A Topological Analysis of Monitor Placement. In IEEE Sixth International Symposium on Network Computing and Applications, July 2007.

[17] R. Kompella and C. Estan. The Power of Slicing in Internet Flow Measurement. In USENIX IMC 2005, October 2005.

[18] M. Krentel. Generalizations of opt p to the polynomial hierarchy. In Theor. Compu. Sci. 97 (2):183-198, 1992.

[19] S. Lee, T. Wong, and H. S. Kim. Secure split assignment trajectory sampling: a malicious router detection system. In IEEE DSN'06, 2006.

[20] A. Mizrak, Y. Cheng, K. Marzullo, and S. Savage. Fatih: Detecting and Isolating Malicious Routers. In DSN, 2005.

[21] A. Mizrak, Y. Cheng, K. Marzullo, and S. Savage. Detecting and Isolating Malicious Routers. In IEEE Transaction on Dependable and Secure Computing, 2006.

[22] Packet Design, Inc. Route Explorer. http://www.packetdesign.com/products/products.htm.

[23] V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. G. Andersen. cSAMP: A System for Network-Wide Flow Monitoring. In USENIX NSDI, 2008.

[24] A. Shaikh and A. Greenberg. OSPF Monitoring: Architecture, Design and Deployment Experience. In USENIX NSDI, 2004.

[25] N. Spring, R. Mahajan, and D. Wetheral. Measuring ISP topologies with RocketFuel. In SIGCOMM, August 2002.

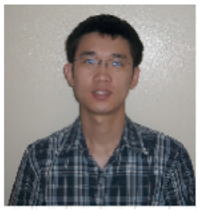
[26] K. Suh, Y. Guo, J. Kurose, and D. Towsley. Locating network monitors: Complexity, heuristics and coverage. March 2005.

[27] J. Winick and S. Jamin. Inet-3.0: Internet topology generator. Technical Report UM-CSE-TR-456-02, University of Michigan, 2002.

[28] H. Zang and A. Nucci. Optimal NetFlow Deployment in IP Networks. In Proc. of International Teletraffic Congress (ITC), Aug 2005.

[29] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In Proceedings of IEEE INFOCOM, March 1996.

[30] B. Zhang. Efficient Traffic Trajectory Error Detection. PhD thesis, Rice University, 2010.



Bo Zhang is a Ph.D. student in Computer Science at Rice University. He received a B.S. in Computer Science in 2004 from the University of Science and Technology of China and a M.S. in Computer Science in 2007 from Rice University. His research interest lies in network management and network measurement.

Guohui Wang is a Ph.D. student in Computer Science at Rice University. He received a B.S. in Electrical Engineering from University of Science and Technology of China in 2002, a M.S. in Computer Science from Chinese Academy of Science in 2005 and a M.S. in Computer Science from Rice University in 2008. His research interests are in networking and distributed systems.

Angela Yun Zhu is a Ph.D. student in Computer Science at Rice University. She received a B.S. in Computer Science from the University of Science and Technology of China in 2003, a M.S. in Computer Science from the University of Science and Technology of China in 2006 and a M.S. in Computer Science from Rice University in 2009. Her research interests are in programming languages.

T. S. Eugene Ng is an Assistant Professor of Computer Science at Rice University. He is a recipient of a NSF CAREER Award (2005) and an Alfred P. Sloan Fellowship (2009). He received a Ph.D. in Computer Science from Carnegie Mellon University in 2003. His research interest lies in developing new network models, network architectures, and holistic networked systems that enable a robust and manageable network infrastructure.