Measurement Methods for Fast and Accurate Blackhole Identification with Binary Tomography

Ítalo Cunha
Thomson / UPMC Paris Universitas
italo.cunha@thomson.net

Renata Teixeira
CNRS / UPMC Paris Universitas
renata.teixeira@lip6.fr

Nick Feamster
Georgia Tech
feamster@cc.gatech.edu

Christophe Diot
Thomson
christophe.diot@thomson.net

ABSTRACT

Binary tomography—the process of identifying faulty network links through coordinated end-to-end probes—is a promising method for detecting failures that the network does not automatically mask (e.g., network "blackholes"). Because tomography is sensitive to the quality of the input, however, naïve end-to-end measurements can introduce inaccuracies. This paper develops two methods for generating inputs to binary tomography algorithms that improve their inference speed and accuracy. *Failure confirmation* is a per-path probing technique to distinguish packet losses caused by congestion from persistent link or node failures. *Aggregation strategies* combine path measurements from unsynchronized monitors into a set of consistent observations. When used in conjunction with existing binary tomography algorithms, our methods identify all failures that are longer than two measurement cycles, while inducing relatively few false alarms. In two wide-area networks, our techniques decrease the number of alarms by as much as two orders of magnitude. Compared to the state of the art in binary tomography, our techniques increase the identification rate and avoid hundreds of false alarms.

Categories and Subject Descriptors

C.2.3 [**Computer Systems Organization**]: Computer Communication Networks—*Network Monitoring*

General Terms

Design, Experimentation, Measurement

Keywords

Network Tomography, Troubleshooting, Diagnosis

1. INTRODUCTION

Binary tomography refers to the process of detecting and identifying link failures by sending coordinated end-to-end probes [10]. This technique offers great hope for network administrators to diagnose failures that are not possible to detect with existing network alarms. One such class of failures is network "blackholes", or failures that network devices and protocols cannot automatically mask. Blackholes may be caused by router software bugs [7], errors in the interaction between multiple routing layers [11], or router misconfigurations [12, 17]. In many cases, end-to-end packet loss or outright loss of reachability may be the only indication that a link or node has failed [16]. Even if a customer notices a failure, operators may not be able to determine its location; moreover, operators would prefer to detect failures before customers complain.

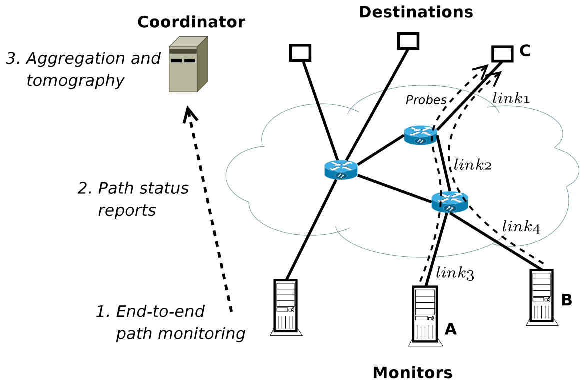


Figure 1: Example of binary tomography.

Fig. 1 shows an example of applying binary tomography to fault diagnosis. *Monitors* periodically probe destinations. A *coordinator* combines the results of probes from all monitors and runs a tomography algorithm. The goal is that when a link fails, a unique set of end-to-end paths from monitors to destinations experiences the failure. Binary tomography algorithms then use the set of paths that experience end-to-end losses to identify the failed link.

Despite its promise, binary tomography algorithms are difficult to apply directly in real networks [14]. Binary tomography has been explored extensively in theory, simulations, and offline analysis [9, 10, 16, 22]. Unfortunately, our experimental results show that applying tomography algorithms to measurements collected from deployed network monitoring systems leads to a large number of alarms, many of which do not correspond to failed links (*false alarms*). For instance, the naïve application of tomography in our measurements from PlanetLab would trigger almost one alarm per minute; in our measurements from an enterprise network, these algorithms would raise one alarm every three minutes. Such alarm rates are much too high to ever be useful in practice.

False alarms arise because the inputs to the tomography algorithm are inaccurate. Binary tomography takes as input the network topology and a set of end-to-end measurements (formalized as a *reachability matrix*, which indicates whether each path is up or down). Building accurate network topologies has already received considerable attention [23, 27]. Because an inconsistent reachability matrix will often lead to false alarms, this paper focuses on building consistent reachability matrices. In the example in Fig. 1, suppose that link 4 is working, but that the reachability matrix views the path between A and C as up and B and C as down; binary tomography would raise a false alarm with link 4 as down.

These inconsistencies arise for two reasons: (1) *detection errors:* when there is no failure, but the path from B to C was mistakenly detected as down; and (2) *lack of synchronization:* link 2 failed, but when A probed C the link was still up. Type 1 errors occur because packet losses are often bursty, and hence monitors can easily misinterpret a transient but bursty loss incident as a persistent blackhole. Type 2 errors arise because it is practically impossible to guarantee that probes issued by different monitors to different destinations will cross a link at the same time. Early tomography algorithms [6] assumed multicast probes from a single monitor to achieve synchronization, but multicast is not widely deployed yet.

This paper develops measurement methods that address both types of errors to build accurate reachability matrices to help existing tomography algorithms quickly detect and locate persistent failures while raising as few false alarms as possible. We evaluate these methods with analytical modeling, controlled experiments, and wide-area measurements. We make the following two contributions:

1. **A probing method for quickly distinguishing persistent path failures from transient congestion.** In Sec. 2, we design and evaluate a probing strategy for *failure confirmation* that distinguishes persistent path failures from transient packet losses. We show that periodic probing minimizes detection errors (i.e., errors of Type 1). We are able to compute the optimal number of probes and interval between probes for achieving a target detection-error rate with minimal overhead.
2. **Strategies for aggregating path failures into a consistent reachability matrix.** In Sec. 3, we develop and validate *aggregation strategies* to reduce Type 2 errors. These strategies introduce a delay to verify that measurements are stable before producing inputs for binary tomography. Our analytical and experimental results show that aggregation strategies are essential to address the impossibility to synchronize measurements.

We apply these confirmation and aggregation methods to evaluate how they can improve the accuracy of existing binary tomography methods. In Sec. 4, we show with controlled experiments that applying our techniques to existing tomography algorithms can quickly and accurately identify all persistent failures with few false alarms. Applying these techniques to end-to-end probes from PlanetLab and an enterprise network reduces the number of alarms by two orders of magnitude. Controlled experiments also show that our techniques identify more persistent failures than the state-of-the-art approach [16], and avoid hundreds of false alarms.

2. DETECTING PATH FAILURES

A lost probe might indicate a persistent failure, but it can also indicate a transient loss due to congestion, routing convergence, or overload at hosts. Assuming that every lost probe indicates a failure could lead to many *false alarms*—cases where the tomography algorithm claims that a link has failed when it has not. In this section, we develop a probing method to distinguish persistent failures from congestion-based transient losses. We refer to this method as *failure confirmation*. We are *not* interested in measuring loss rates; rather, making a binary decision allows us to develop mechanisms with lower probing overhead and delay than mechanisms that measure loss rates [25]. Although one of our motivations is to diagnose blackholes, our techniques detect a broader class of failures that includes blackholes as well as other types of failures that could be detected in other ways.

We aim to detect failures as quickly as possible while reducing the overall number of *detection errors*—cases where we misclassify a transient loss as a failure. Additional probing can reduce detection errors at the cost of increasing the time to detect a failure (perhaps by as much as tens of seconds, depending on the overall probing rate). The rest of this section examines the probing process, rate, and number of probes that allows for fast detection and low overall detection-error rate for the types of losses that occur on Internet paths.

2.1 Confirmation Method

When a monitor observes a single lost probe along a path, it sends additional failure confirmation probes to determine whether lost packets are caused by a failure. The goal of failure confirmation is to determine whether the path has failed or is simply experiencing transient packet loss. A *confirmed failure* happens when all confirmation probes are lost.

We model path losses using the *Gilbert model* [13], an accurate model for capturing burstiness of congestion-based losses observed on Internet paths [29]. In a Gilbert model, paths are either in a good state, where all transmissions are successful; or in a bad state, where all packets are lost. This model has two parameters: the probability to transition from the bad to the good state and the probability to transition from good to bad.

Detection errors are unavoidable in real deployments, and we would need to send an infinite number of confirmation probes to achieve perfect detection. Our objective is to make the detection-error rate, F, as small as possible while still keeping detection time low. We define κ as the number of confirmation probes and T as the time to run failure confirmation. In this loss model, we denote the average length of loss bursts by b and the average loss rate on the path by r. Tab. 1 summarizes the notation used in the paper.

We first show that a periodic probing process minimizes F, given a fixed κ and T. The second part of our analysis assumes periodic probing and takes as input a target F. When the number of probes, κ, is too large, probes will interfere with the network (perhaps inducing additional losses); when κ is too small, detection errors increase. The objective is to find values of κ and T that achieve the target F.

2.1.1 *Probing process*

We show that a periodic probing process minimizes the detection-error rate, F, given κ and T. Minimizing detection errors is equivalent to minimizing the probability that all confirmation probes fall within loss bursts. In the Gilbert model [13], the probability of losing a confirmation probe at time t_i given that a probe was lost at time t_{i-1} is:

$$\Pr(\,\text{loss}(t_i)\,|\,\text{loss}(t_{i-1})\,) = r + (1-r)e^{-\mu_i/b},$$

where $\mu_i = t_i - t_{i-1}$ is the time interval between probe i and $i-1$. Thus, given κ confirmation probes, we can express the detection-error rate as:

$$F = \prod_{1 \le i < \kappa} r + (1-r)e^{-\mu_i/b}. \qquad (1)$$

We can then find the intervals between probes, $\mu_1, \cdots, \mu_{\kappa-1}$, that minimize F by solving an optimization problem constrained by the total time available to run confirmation, i.e., $\sum_\kappa \mu_i < T$. The optimal solution for this optimization occurs when all μ_i are equal. One way to prove this result is by showing that if there exists $\mu_i > \mu_j$, then decreasing μ_i by δ and increasing μ_j by δ decreases the value of F. Thus, equally-spaced probes are the best strategy to minimize detection errors. Bolot *et al.* [5] have proved a similar result in the context of FEC for VoIP.

Although sending periodic probes minimizes the detection-error rate if losses follow a Gilbert model, this method performs poorly in the unlikely case of periodic losses. To avoid the possibility of *phase locking*, i.e., losing all confirmation probes in periodic loss episodes if μ is a multiple of the loss period, we use the method suggested by Baccelli *et al.* [1], where probe inter-arrival times are uniformly distributed between $[(1-\gamma)\mu,\ (1+\gamma)\mu]$, with $0 \le \gamma < 1$.

We extend Eq. (1) to consider these uniform inter-arrival times by calculating F as a function of γ:

$$F = \left[r + (1-r)e^{-\mu/b} \frac{b}{2\gamma\mu} \left(e^{\gamma\mu/b} - e^{-\gamma\mu/b} \right) \right]^\kappa. \qquad (2)$$

Eq. (2) is an extended Eq. (1) with an extra term that is larger than one and increases with γ, capturing the increased probability of two confirmation probes falling in the same loss burst if their inter-arrival time is less than μ.

Proofs and detailed explanation of the results in this subsection are available in an extended version of this paper [8].

2.1.2 Number of probes and rate

We now discuss how to set κ and T to achieve a target F assuming confirmation probes have inter-arrival times between $[(1-\gamma)\mu,\ (1+\gamma)\mu]$. We can express T as $\kappa \times \mu$, where μ is the average interval between probes. We formulate two optimization problems for selecting κ and μ:

Minimizing time. The first optimization model minimizes the total time T it takes to run the confirmation scheme, subject to the target F (constraint a) and a maximum probing rate of $1/\mu_{\min}$ packets per second (constraint b):

$$\begin{aligned} \min \quad & \kappa \times \mu & (3) \\ \text{s.t.} \quad & \kappa \times \ln\left(r + (1-r)e^{-\mu/b} f(\mu)\right) < \ln(F) & (a) \\ & \mu > \mu_{\min} & (b) \end{aligned}$$

where $f(\mu) = b(e^{\gamma\mu/b} - e^{-\gamma\mu/b})/2\gamma\mu$. The intuitive solution to this optimization problem is to send probes often (i.e., μ is close to $\mu_{\min}$) and increase the number of confirmation probes until F is achieved.

Minimizing probes. The second optimization model minimizes the total number of confirmation probes needed to achieve F. Assuming independent losses, we can drop the second term of Eq. (2) and express the probability of detection errors as $F = r^\kappa$. We can then compute the number of confirmation probes and inter-probe spacing with:

$$\begin{aligned} & \kappa = \lceil \ln(F)/\ln(r) \rceil, \text{and} & (4) \\ \min \quad & \mu & \\ \text{s.t.} \quad & \kappa \times \ln\left(r + (1-r)e^{-\mu/b} f(\mu)\right) < \ln(F) & (a) \\ & \mu > \mu_{\min} & (b) \end{aligned}$$

where $f(\mu) = b(e^{\gamma\mu/b} - e^{-\gamma\mu/b})/2\gamma\mu$. This μ is usually much higher than $\mu_{\min}$ because it must support the assumption of independent losses.

Any other combination of κ and μ achieving F would be an intermediate solution between these two extremes. For the rest of this paper, we use the approach in Eq. (4) to set the parameters of κ and μ.

Var.	Description
Confirmation Scheme	
κ	Number of confirmation probes
μ	Average interval between confirmation probes
$\mu_{\min}$	Minimum interval between conf. probes
γ	Varies interval between conf. probes in $(1 \pm \gamma)\mu$
F	Target detection-error rate
T	Total time running confirmation ($\kappa \times \mu$)
r	Average packet loss rate
b	Average length of loss bursts
Aggregation Strategies	
C	Measurement cycle duration
$\mathcal{P}$	Set of monitored paths
$\mathcal{H}_\ell$	Hitting set (paths going through link ℓ)
H	Average hitting set size
f	Failure length
n	Number of cycles in aggregation
N	Average number of paths in $\mathcal{H}_\ell$ probed in a cycle
w	Average number of detection errors in a cycle
q	Fraction of matrices built due to detection errors
t_{de}	Time to detect a failure

Table 1: Notation.

2.1.3 Deriving parameters in practice

Computing κ and μ requires five parameters. The operator selects the desired target detection-error rate (F), the minimum probing interval ($\mu_{\min}$), and the variability in probe inter-arrival times (γ). The path loss rate (r) and average burst length (b) are properties of the paths.

Estimating path loss rate and burst length. Path packet loss rate can be measured with a plethora of tools ranging from router-based measurements [3, 21], to active probing [25, 26], to passive monitoring of application traffic [20]. One can use the technique proposed by Sommers *et al.* [25] from the set of monitors to estimate the values of r and b. Given that these values will vary over time, we suggest that operational deployments perform multiple measurements and pick a value slightly above the maximum loss rate and loss burst values. Overestimated values of r and b make confirmation more robust to estimation errors.

Selecting target detection-error rate, minimum probing rate, and inter-arrival time variability. The target detection-error rate, F, should be small to allow tomography algorithms to operate on relatively accurate estimates of path failures. However, choosing F too small increases κ, thereby increasing delay, and reducing F gives diminishing returns (as we see in Sec. 2.2.2). In this paper, we select F experimentally.

The value of $\mu_{\min}$ should be set so that the probes are not intrusive; previous work has shown that bursts as short as 10 packets can affect router queues during loss periods [25]. One way to mitigate this effect is to limit the number of confirmation probes queued at routers to a very small number. Current router configurations use 180 ms of buffer [4]; as an example, we can set $\mu_{\min} = 100$ ms to limit the maximum number of confirmation packets in router queues to two.

The variability of inter-arrival times should be large enough to avoid phase locking between probes and periodic losses. In the worst-case scenario where the loss period equals μ, choosing $\gamma = xr$ implies that the probability of sending a probe in a loss episode is less than $1/2x$. Thus, small values of γ are enough to avoid phase locking, resulting in limited impact on F. In the rest of this paper, we use $\gamma = 0.1$, which is applicable to a wide range of loss rates [1].

2.2 Evaluation

We evaluate our probing method in a controlled environment using Emulab and in wide-area experiments using PlanetLab. Controlled experiments allow us to measure the accuracy of our technique, because we have ground truth. On the other hand, wide-area experiments allow us to test our method under actual loss scenarios.

2.2.1 Controlled experiments

Because Emulab can only introduce random losses, we deployed a modified version of the Linux Netem module that allows us to inject random (i.e., Poisson), bursty (i.e., up to three-state Gilbert models), and periodic losses on links. In this section, we use Fast Ethernet links between nodes, with packet losses occurring in both directions independently. We used the Abilene OSPF topology and varied emulated link latencies from zero (i.e., only native OS and hardware delays) to 20 ms, with quantitatively similar results. We add background traffic of 500 packets per second in each direction to trigger state changes in the Gilbert model when emulating bursty losses. To study our confirmation scheme with different configurations, we also varied the values of κ and μ. We set $F = 10^{-5}$, as this value represents a good trade-off for a wide range of path loss rates, and $\mu_{\min}$ to 100 ms, which guarantees that at most two confirmation probes are simultaneously queued at routers [4].

We investigated how detection errors vary as a function of path loss rates and burst lengths. To cover a large portion of previously reported loss rates [20, 21, 25, 26], we varied path loss rates between 0.01% and 1% and average burst lengths from 4 ms to 40 ms. We used synthetic and real ISP topologies and found qualitatively similar results.

Detection-error rates with different loss rates. In our experiments, detection errors increase when r increases and κ is kept constant. Our proposed scheme varies κ between two and five to achieve the target $F = 10^{-5}$, depending on the path loss rate. Spacing probes significantly reduces detection errors compared to sending back-to-back probes. If probes are sufficiently spaced, detection errors decrease with diminishing returns when r is constant and κ increases, as expected from Eq. (2).

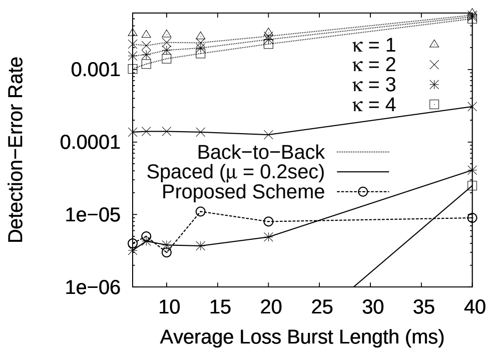


Figure 2: Burst length vs detection errors—Emulab.

Detection-error rates with different burst lengths. Fig. 2 shows the detection-error rate when varying the average burst length in the Abilene topology for fixed per-link loss rates of 0.1%. Results for other loss rates are qualitatively similar. For $\kappa \geq 2$, we show results for back-to-back probes and probes spaced by 200 ms. For back-to-back probes, the detection-error rate increases with the burst length, as the probability of confirmation packets falling in the same loss burst increases. Results for $\mu = 200$ ms are independent of the burst length (i.e., a straight line) for $b \leq 20$ ms because the probability of a loss burst lasting more than 200 ms in these cases is close to zero. However, when $b = 40$ ms, the probability of a burst lasting more than 200 ms is higher and the detection-error rate increases. Sending two spaced confirmation probes is better than sending many back-to-back ones. Finally, the proposed scheme is the only one that achieves the target detection-error rate of 10^{-5} for all burst lengths, by setting $\kappa = 4$ and varying μ between 100 ms and 398 ms. When bursts are long, μ is increased to avoid sending probes during the same loss burst. When bursts are short, μ is equal to $\mu_{\min}$ to reduce total confirmation time. When $b = 4$ ms, the total confirmation time ($T = \kappa \times \mu$) of the proposed scheme is 400 ms, which is half the confirmation time when using $\kappa = 4$ and $\mu = 200$ ms (i.e., squares with solid line in Fig. 2).

Detection-error rates with other loss models. We evaluated the effect of applying our method to losses given by a more general three-state Gilbert model, capable of capturing loss processes where the probability of losing a single packet is different from having a loss burst. We found that the increase in F is less than one order of magnitude even if the fraction of single packet losses (i.e., no burst) is as high as 90%, a scenario where the two-state model is very inaccurate. We also ran experiments with periodic losses and saw that varying packet inter-arrivals with $\gamma > r$ is enough to prevent phase locking. We present a more detailed discussion in an extended version of this paper [8].

Summary. These experiments show that our probing scheme adapts the value of κ and μ to path loss rates and burst lengths to successfully achieve a target detection-error rate while minimizing total confirmation time. The method is also robust to different loss processes.

2.2.2 Wide-area Internet experiments

We aim to determine whether failure confirmation is useful in practical scenarios. Our deployment in an enterprise

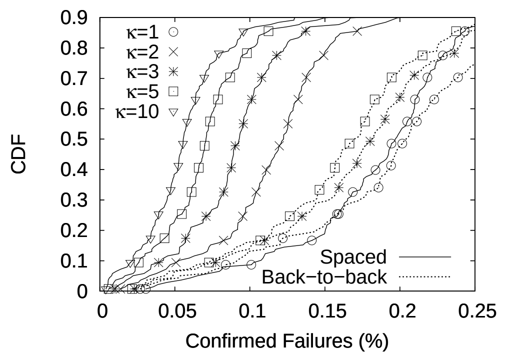


Figure 3: Efficacy of confirmation in PlanetLab.

network resulted in only a few iterations of the confirmation method, because this network is fairly stable and we only ran the experiment for two weeks. We then used a larger deployment with 200 PlanetLab nodes, each of which probed the other nodes periodically. PlanetLab nodes are frequently overloaded and induce short bursts of packet loss [24]. Although we cannot differentiate host- and network-induced packet loss (a situation that can happen in any real deployment), PlanetLab's dynamic environment is very demanding of confirmation methods and worth investigating.

We compared the relative performance of two confirmation schemes run during the same time period, so that they will experience similar node conditions. The first scheme sends confirmation probes back-to-back, while the second sends confirmation probes spaced by 200 ms. If back-to-back probes confirm a failure, but spaced probes do not, we know that the first has raised a detection error. We pick $\mu = 200$ ms based on the observation that incidence of bursty packet loss usually lasts less than 100 ms [18,21,25]. We also used confirmation probes spaced by 2 seconds and found similar results.

Figure 3 shows the cumulative distribution of the fraction of confirmed failures for each PlanetLab node, for different numbers of confirmation probes, and for different probing strategies. We compute the fraction of confirmed failures by dividing the number of confirmed failures by the total number of lost probes at each node. We see that increasing κ reduces the number of confirmed failures (i.e., detection errors). The two rightmost lines show the fraction of confirmed failures when $\kappa = 1$. A larger κ shows that sending probes with delay in between each probe confirms significantly fewer failures than the one using back-to-back probes. We also see diminishing returns as in the controlled experiments. When probes are spaced, increasing κ from five to ten decreases detection errors by (1.3%) less than increasing κ from three to five (1.8%). Using more than ten confirmation probes yields minimal improvement. When using five confirmation probes, spacing probes confirms 38% less failures than using back-to-back probes (i.e., the difference between the two curves for $\kappa = 5$). All failures in these 38% are detection errors, indicating that using the proposed scheme can significantly improve the quality of measurements to be used by tomography algorithms.

3. AGGREGATING FAILURES

This section develops *aggregation* methods for combining path measurements into a reachability matrix (Step 3 in Fig. 1). The main challenge with aggregation is constructing a reachability matrix where the end-to-end path measurements have a *consistent* view of the status of the links in the network. An inconsistent reachability matrix could introduce false alarms or wrong identification of failures. Existing work on binary tomography algorithms presumes that the reachability matrix is consistent; in practice, however, consistency is difficult to achieve. This section explains the challenges in achieving consistency, proposes strategies for constructing a consistent reachability matrix, analyzes their delay, and evaluates them both analytically and empirically.

3.1 Definitions

We define a *reachability matrix* as a matrix, M, where each entry M_{md} is the binary status (i.e., up or down) of the path from monitor m to destination d. Note that the matrix may be incomplete, since not all monitors will monitor all destinations. We define *aggregation* as the process of combining path status measurements from the monitors to construct this reachability matrix. Suppose that some set of monitored paths cross a particular link in the network. Informally, *consistency* says that if a particular link is "down" then all paths that cross that link should have status "down", and that if all links in a path are "up" then that path should have status "up". We now formalize this notion.

Let $\mathcal{P}_m$ denote the set of paths probed by a monitor m and $\mathcal{P} = \cup_{\forall m}\mathcal{P}_m$ the set of paths from all monitors. We define the *hitting set* of a link ℓ, $\mathcal{H}_\ell$, as the set of paths that traverse ℓ.[1] We say that a reachability matrix is *consistent* at an instant in time if, for every ℓ that is failed, all paths in $\mathcal{H}_\ell$ have a status of down and all paths that contain only working links have a status of up in the reachability matrix.

3.2 Challenges in Achieving Consistency

Binary tomography algorithms take as input a consistent reachability matrix; unfortunately, two factors make it difficult to construct a consistent reachability matrix in practice:

Lack of synchronized measurements. Because monitors probe at different times, different monitors may observe different characteristics for the same link, thus creating inconsistencies in the reachability matrix.

Our goal is to design an aggregation strategy that builds a consistent reachability matrix with small aggregation delay. If monitors could probe all paths in $\mathcal{P}$ simultaneously, then the resulting reachability matrix would be consistent. Many tomography algorithms [2, 9, 30] assume consistent inputs. Unfortunately, synchronous measurements are impossible in practice. First, measurements from a single monitor are not instantaneous, because probing many destinations takes time (in our experiments, this process takes from tens of seconds to minutes). Second, each monitor probes a different set of paths, so it is impossible to guarantee that two monitors probe the same link simultaneously. Monitors have different cycle lengths as each monitor probes a different set of destinations, and machines have different processing power and available bandwidth. The overall *cycle length*, C, is the time it takes the slowest monitor to probe all its paths; as we will see in Sec. 3.4, the overall aggregation delay is a function of both the cycle length and the aggregation strategy.

Detection errors. Detection errors from failure confirma-

[1]For simplicity, we refer to the hitting set of a link, but this definition also applies for sets of links.

tion (Sec. 2) may create situations where a path is considered to have failed when it has not. These errors in the reachability matrix may also cause a tomography algorithm to reach incorrect conclusions about which links have failed. In this section, we use F to denote the *actual* detection-error rate of failure confirmation, which may be different from the *target* detection-error rate of Sec. 2.

3.3 Aggregation Strategies

We propose and evaluate three strategies for aggregating path measurements into a reachability matrix. The first method is simple and fast, but it can build inconsistent matrices if failures are short or if there are detection errors (BASIC, Sec. 3.3.1). We then consider two enhancements that wait longer to build matrices but achieve higher consistency: a conservative one (MC, Sec. 3.3.2) and another that is more tolerant to detection errors (MC-PATH, Sec. 3.3.3). We present models that capture how detection errors and unsynchronized measurements may introduce inconsistency.

3.3.1 *Basic approach*

The BASIC aggregation strategy works as follows. First, detect that a path status changed from up to down. Then, wait a full cycle C for monitors to probe all paths in $\mathcal{P}$. Finally, build a reachability matrix by combining the path statuses reported in the latest measurement cycle. This simple strategy assumes that if ℓ fails, all paths in $\mathcal{H}_\ell$ will be confirmed as down after C.

Consistency analysis. The consistency of BASIC depends on the duration of the failure, f, relative to the cycle length, C. We analyze consistency in three scenarios. We first assume that failure confirmation gives no detection errors, and relax this assumption later.

Scenario 1 (long failures): $f > 2C$. In S1, monitors probe all paths in $\mathcal{H}_\ell$ while ℓ is down; hence, the coordinator always builds a consistent reachability matrix. The average consistency in this scenario is 1.

Scenario 2 (intermediate failures): $C \le f \le 2C$. In S2, all paths in $\mathcal{H}_\ell$ will be down during a full cycle because the failure is longer than C. When the coordinator builds the reachability matrix, however, ℓ may have recovered. In this case, the matrix is consistent with ℓ's failure, but the failure no longer persists. We call these instances *late identifications.* We consider these cases consistent, but it is simple to extend the analysis to consider late identification as inconsistent, but we omit this analysis for conciseness.

Scenario 3 (short failures): $f < C$. In S3, the coordinator may build an inconsistent reachability matrix because monitors may probe some paths in $\mathcal{H}_\ell$ while ℓ is down and others when it is up. Let $\mathcal{F}$ be the set of all possible failures and $H = \sum_{\ell \in \mathcal{F}} |\mathcal{H}_\ell|/|\mathcal{F}|$ be the *average hitting set size.* We identify two cases:

1. Late identification: The probability of probing all paths in $\mathcal{H}_\ell$ while ℓ is failed and getting a consistent reachability matrix is approximately $p = (f/C)^H$.

2. Inconsistent reachability matrix: In all other cases, the reachability matrix will be inconsistent. The consistency of the reachability matrix in these cases depends on the number of paths in $\mathcal{H}_\ell$ that were probed during f. Given that the reachability matrix is inconsistent, at least one path has to probe ℓ during f (i.e., the path that detected the failure) and at most $\mathcal{H}_\ell - 1$ can probe ℓ during f (otherwise, the matrix would be consistent). Hence, we can approximate the average number of paths in $\mathcal{H}_\ell$, for all $\ell \in \mathcal{F}$, probed during a failure by $N = 1 + (f/C) \times (H-2)$. The average consistency in these cases is $1 - N/|\mathcal{P}|$.

Combining these three scenarios, the expected consistency of BASIC when there are no detection errors is:

$$E[\text{cons}_{\text{basic}}] = \begin{cases} 1 & \text{if } f \ge C, \\ p + (1-p)\left(1 - \frac{N}{|\mathcal{P}|}\right) & \text{if } f < C. \end{cases} \quad (5)$$

Detection errors reduce consistency. We derive an upper bound for the consistency of BASIC when there are detection errors as follows:

$$E[\text{cons}_{\text{basic, de}}] < q\Big(1 - \frac{w}{|\mathcal{P}|}\Big) + (1-q)E[\text{cons}_{\text{basic}}]. \quad (6)$$

The first term on the right-hand side accounts for matrices constructed due to detection errors only (i.e., not related to any failures), and q is the fraction of such matrices. Their average consistency is $1 - \frac{w}{|\mathcal{P}|}$, where $w = 1 + F(|\mathcal{P}| - 1)$ is the average number of paths with wrong status in cycles where detection errors occur. The second term corresponds to matrices built due to real failures. Their average consistency is $E[\text{cons}_{\text{basic}}]$ if there are no detection errors. If there are detection errors during aggregation of real failures, consistency will be lower than $E[\text{cons}_{\text{basic}}]$, which is why the equation is an upper bound.

Unfortunately, BASIC may produce incorrect answers due to either lack of synchronization if $f < C$ or detection errors. Thus, we also considered other aggregation strategies that run in one measurement cycle: (1) building the reachability matrix every C seconds, (2) waiting a window of time without path status changes to build the reachability matrix [14], and (3) continuously updating the reachability matrix as monitors report new measurements. All of these techniques create inconsistent matrices if $f < C$ or if detection errors occur. In the next two sections, we describe improvements to consistency in the face of these two challenges.

3.3.2 *Coping with lack of synchronization*

Multi-cycle aggregation strategies improve consistency by using extra time to gather more measurements and build a more stable reachability matrix. Similar to basic aggregation, the *multi-cycle* (MC) strategy starts aggregation upon a path status change. Instead of building the reachability matrix after one measurement cycle, MC waits for n cycles with identical measurements to build the reachability matrix. MC avoids building reachability matrices when measurements are not stable and achieves the highest consistency of the three approaches, as we show below.

Consistency analysis. Suppose that there are no detection errors. There are four scenarios: (1) When ℓ fails for $f > (n+1) \times C$, monitors obtain a stable status for $\mathcal{H}_\ell$ during n cycles and MC builds a consistent matrix. (2) When $f < (n-1) \times C$, monitors do not observe $\mathcal{H}_\ell$ as down for n cycles and no matrix is built. (3) When ℓ fails for $n \times C < f < (n+1) \times C$, ℓ is down for n cycles, but it may recover before MC builds the reachability matrix, thereby leading to late identification (as in S2). (4) When ℓ fails for $(n-1) \times C < f < n \times C$, monitors may probe some paths in

$\mathcal{H}_\ell$ in n cycles and others in $n-1$ cycles. If monitors probe all paths in $\mathcal{H}_\ell$ in n cycles, then there is a late identification. Otherwise, measurements at the n^{th} cycle are different from the $n-1$ previous cycles and no matrix is built. Putting all cases together, MC always builds consistent matrices:

$$E[\text{cons}_{\text{mc}}] = \begin{cases} 1 & \text{if } f \geq (n-1) \times C, \\ \text{none} & \text{if } f < (n-1) \times C. \end{cases} \quad (7)$$

MC is robust to detection errors. To be aggregated, detection errors must occur on the same set of paths for n consecutive cycles. If there are frequent detection errors, though, aggregation must wait until n consecutive matrices are identical. This additional delay affects the number of failures that MC can build a matrix for. Thus, MC is less useful in the face of detection errors.

3.3.3 *Coping with detection errors*

We propose an extension to MC called *multi-cycle noise-tolerant* (MC-PATH), which achieves lower consistency but is more tolerant to detection errors. MC-PATH also waits n cycles to build the reachability matrix. But, instead of requiring the statuses of all paths in $\mathcal{P}$ to be identical, paths that are down for n consecutive cycles are added as down in the reachability matrix, while others are added as up. In other words, only paths down in the last n cycles are aggregated as down, while unstable paths are aggregated as up.

Consistency analysis. The consistency of MC-PATH is the same as that of MC, except for failures where a link ℓ fails for $(n-1) \times C < f < n \times C$. In this case, monitors may probe paths in $\mathcal{H}_\ell$ during n or $n-1$ cycles. Similarly to S3 of BASIC, MC-PATH builds an inconsistent matrix with only the paths that were down in n cycles. The average consistency of MC-PATH when there are no detection errors is then:

$$E[\text{cons}_{\text{path}}] = \begin{cases} \text{none} & \text{if } f < (n-1)C, \\ 1 & \text{if } f \geq n \times C, \\ p' + (1-p')\left(1 - \frac{N'}{|\mathcal{P}|}\right) & \text{otherwise.} \end{cases} \quad (8)$$

where $p' = (f'/C)^H$, $N' = 1 + \frac{f'}{C} \times (H-2)$, and $f' = f \bmod C$ is the length of the failure in the n^{th} cycle.

Detection errors are filtered on a per-path basis and do not delay the aggregation of paths in $\mathcal{H}_\ell$. When there are detection errors, an upper bound on the consistency of MC-PATH would be similar to Eq. (6). The average number of paths with incorrect status, w, is $1 + F^n(|\mathcal{P}| - 1)$, and q is much smaller because the probability of building a reachability matrix due to detection errors only is proportional to F^n instead of F. For MC-PATH, the first term of Eq. (6), which represents inconsistency caused by detection errors, approaches zero as n increases. Hence, we have:

$$\lim_{n\to\infty} E[\text{cons}_{\text{path, de}}] = E[\text{cons}_{\text{path}}]. \quad (9)$$

3.4 Aggregation Delay

The *aggregation delay* of all aggregation strategies depends on the time it takes to detect the failure, t_{de}, the cycle length, C, and the number of cycles aggregation waits before building the matrix, n. Apart from MC, which can take arbitrarily long to build matrices due to detection errors, the average delay can be written as:

$$E[\text{delay}] = E[t_{\text{de}}] + n \times C, \quad (10)$$

where $n = 1$ in BASIC. The detection time $E[t_{\text{de}}]$ is smaller than C because all paths are probed in a cycle. Also, $E[t_{\text{de}}]$ is minimized for periodic probing and decreases as $\mathcal{H}_\ell$ increases. We present a model for $E[t_{\text{de}}]$ in an extended version of this paper [8].

Eq. (8) implies that a system can build consistent matrices for failures longer than $n \times C$ and Eq. (10) implies that these failures are identified before $(n+1) \times C$. E.g., an operator who wants to identify failures of a target duration f_{target} (e.g., 5 minutes), should configure the system such that:

$$(n+1) \times C < f_{\text{target}}. \quad (11)$$

Failure to satisfy this condition (e.g., very long cycles or need for large n due to high detection-error rate) means that identifications will be late and that tomography algorithms will use inconsistent reachability matrices.

3.5 Validation

To validate our models of consistency for each approach to aggregation, we perform controlled experiments in Emulab. We also apply our aggregation techniques to measurements collected on both the PlanetLab testbed and across a geographically distributed enterprise network to check how useful they are in practical scenarios.

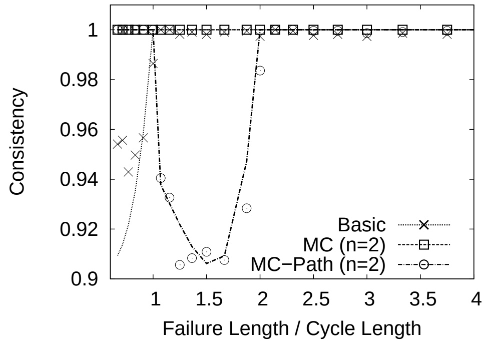
3.5.1 *Controlled experiments*

We use the same Emulab setup as in Sec. 2.2. We assume that any router in the topology can be a destination and consider two different sets of monitors: (1) with only the two farthest routers, at New York and Los Angeles; and (2) with three central monitors at Houston, Chicago, and Salt Lake City. Then, we select the set of paths to probe per monitor, $\mathcal{P}_m$, by using the monitor selection algorithm proposed by Nguyen and Thiran [19]. This algorithm selects the minimum number of paths that allow diagnosis of multi-link failures.

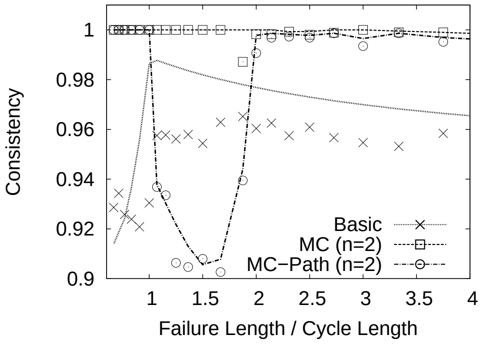
To inject blackholes, we introduce single-link failures that last for 30 seconds and vary the cycle length from 3 to 50 seconds, which will span values of f/C from 0.6 to 10. We also considered more realistic failures from IS-IS message traces collected on Abilene and found qualitatively similar results. We inject congestion-induced packet losses following a Gilbert model. We vary the per-link loss rates between zero and 1% and the burst factor between 4 and 40 ms. These parameter settings correspond to the range of values found by many previous studies in the wide-area [20,25,26]. We only show results for a 1% link loss rate and burst factor of 40 ms, which are more severe than have been observed in practice; results for other configurations are similar. As in Sec. 2, we configure the failure confirmation scheme using $F = 10^{-5}$ and $\mu_{\min} = 100$ ms. Solving Eq. (4) yields $\kappa = 4$ and $\mu = 398$ ms.

Fig. 4 shows the average consistency of each aggregation strategy as f/C varies. Points are the Emulab results and lines are computed from the models in Sec. 3.3. Multi-cycle strategies are running with $n = 2$, so failures need to persist for at least two cycles to be included in the reachability matrix. Fig. 4(a) has no detection errors and Fig. 4(b) has 0.6% incorrect detections (we achieve this high detection-error rate by using only one confirmation probe instead of four).

Although the average consistency is above 90% for all strategies, even small differences in consistency can be significant for tomography algorithms; as shown in Fig. 1, one



(a) No detection errors



(b) 0.6% wrong confirmations

Figure 4: Consistency of aggregation strategies.

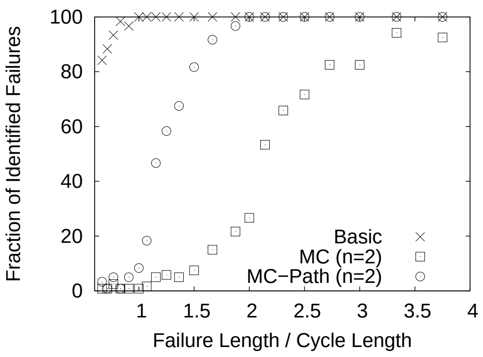


Figure 5: Identified failures, 0.6% wrong detections.

		n				
		1	2	3	4	5
BASIC	Delay	2.51				
	Cons.	0.914				
MC	Delay		15.2	19.0	19.7	22.4
	Cons.		0.981	0.998	1	1
MC-PATH	Delay		6.92	10.0	13.0	16.0
	Cons.		0.964	0.977	0.982	0.985

Table 2: Relationship between aggregation delay (in seconds) and consistency, no confirmation.

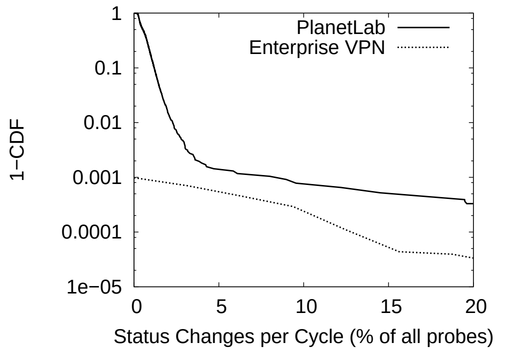
path with incorrect status may trigger a false alarm. We study the effect of aggregation strategies on tomography's accuracy in Sec. 4. High values of consistency occur because the average hitting set size, H, is much smaller than the total number of paths, $|\mathcal{P}|$.

Model validation. Fig. 4 shows that our analytic models of consistency accurately predict the results from the controlled experiments. When there are no detection errors and $f \geq 2C$, all strategies have perfect consistency. MC has perfect consistency in all scenarios. BASIC builds inconsistent matrices when $f < C$, which corresponds to S3. Similarly, MC-PATH creates inconsistent matrices when $1 < f/C < 2$ (recall that $n = 2$ in these experiments). We can explain the shape of MC-PATH curve for $1 < f/C < 2$ from Eq. (8). When the failure ends at the beginning of the n^{th} cycle (f is just a little more than C), the probability of building a consistent reachability matrix, p', is small, but the number of paths with wrong status, N', is also small; hence, consistency is high. When f is almost $2C$, p' is high and the majority of failures will have a consistent matrix. The average consistency is lowest for failures that end in the middle of the n^{th} cycle ($f = 1.5 \times C$), because the number of paths with wrong status is significant and the probability of constructing a consistent matrix is moderate.

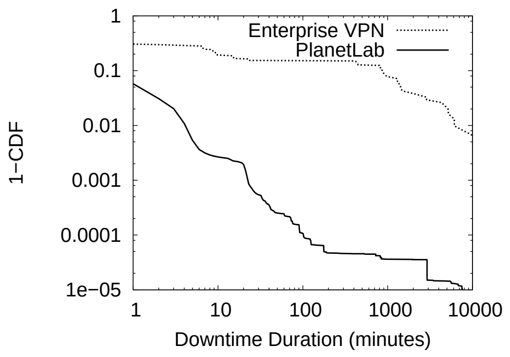
Coping with detection errors. Fig. 4(b) shows that multi-cycle aggregation strategies help mitigate the effects of detection errors. However, BASIC has low consistency because every detection error appears as a path down in the reachability matrix, so consistency is low even when the failure persists for more than three cycles. The analytical bound of BASIC converges to the actual average when f/C is large because the value of q in Eq. (6) approaches one.

Multi-cycle strategies achieve high consistency even in dynamic environments where detection-error rates are high, but these strategies take longer to build a reachability matrix and fail to identify shorter failures. Fig. 5 plots the fraction of the injected failures for which each aggregation strategy builds a consistent reachability matrix when varying f/C. The fraction of identified failures for multi-cycle strategies is low when $f < 2C$ because these strategies cannot obtain stable measurements, so they will not build a reachability matrix, and thus cannot identify these failures. BASIC, on the other hand, always builds a reachability matrix for failures that last more than one cycle, but it also builds many other inconsistent matrices (hence the low average consistency in Fig. 4(b)), which would trigger false alarms. When there are many detection errors, MC misses many failures by being too conservative. It can only build consistent reachability matrices for 27% of the failures that last two cycles. MC-PATH represents the best compromise when there are detection errors, because it builds consistent matrices for all failures that are longer than two cycles (Fig. 5) while keeping the false alarms low (Fig. 4(b)).

Relationship between delay and consistency. Tab. 2 shows the tradeoff between aggregation delay and consistency. We do not use confirmation to focus on delay and consistency of aggregation alone. The BASIC strategy has the lowest consistency but the shortest delay. Multi-cycle strategies can increase delay to achieve higher consistency. MC has the highest consistency, at the cost of the highest delay and missing most of the failures (Fig. 5). Finally, MC-PATH also improves consistency by adding delay. Its consistency is



(a) Matrix Dynamicity



(b) Path Dynamicity

Figure 6: Dynamics of real networks.

lower than MC's, but its delay increases linearly with n and it is able to identify all failures longer than $n \times C$ (Fig. 5).

3.5.2 Wide-area experiments

We use measurements from PlanetLab and an enterprise network to check whether networks are stable enough for BASIC or MC to work and whether path failure events are longer than the typical cycle length in such systems. We do not evaluate consistency directly because we do not have ground truth about failures. The PlanetLab measurements are the same as in Sec. 2, and the enterprise network measurements are collected from eight monitors at sites of the enterprise network across 5 different countries.

Fig. 6(a) shows the complementary cumulative distribution function of the percentage of paths that change status during one measurement cycle. In PlanetLab, cycles last 60 seconds and at least one out of the 39,800 monitored paths changes status every cycle. In this dynamic environment, MC would never build a reachability matrix. In the enterprise network, cycles last 5 seconds and all 56 monitored paths are stable 99.9% of the cycles. Fig. 6(b) shows the complementary cumulative distribution function of the duration of path down events. In PlanetLab, paths stay down for very short periods (95% of paths stay down for only one cycle). These periods are likely caused by overloaded PlanetLab nodes, which may not respond to probes [24]. The fraction of long downtimes in the enterprise network is significantly higher than in PlanetLab. The enterprise network has only few path status changes; when they happen, they are more likely associated to a real failure than in PlanetLab.

We now study which aggregation strategy is more appropriate for each network. PlanetLab has many failures that last less than one cycle and all cycles have path status changes; BASIC would generate many false alarms and MC would never identify a failure (it would wait indefinitely for measurements to be stable). The best for PlanetLab is MC-PATH with large n (e.g., 10 cycles, allowing the identification of all failures longer than 11 minutes as $C = 1$ minute). MC-PATH with $n = 10$ would only generate false alarms for failures that last between 9 and 10 minutes, which are rare (Fig. 6(b) shows that less than 0.015% of path down events last for 9 or 10 minutes). MC-PATH would effectively remove the noise created by unstable paths. In the enterprise network, MC achieves high consistency without compromising the ability to identify long failures because detection errors and short failures are rare.

4. PUTTING IT ALL TOGETHER

This section first shows the results of controlled experiments that evaluate the benefits of using the confirmation and aggregation methods from the previous two sections to produce reachability matrices for binary tomography. Then, it shows that our techniques drastically reduce the number of alarms in PlanetLab and the enterprise network. Finally, we compare our results to the state-of-the-art algorithm introduced by Kompella *et al.* [16] and show that we achieve a higher identification rate with a lower false-alarm rate.

4.1 Setup

We evaluate the effect of our techniques when applied to a simple binary tomography algorithm that uses measurements from multiple monitors [9, 16]. Given a reachability matrix and the network topology, this algorithm uses a greedy heuristic to build a *hypothesis set*, i.e., the most likely set of failed links. First, it creates a candidate set of possibly failed links with all links in failed paths minus all links from working paths. Then, it iteratively selects from the candidate set the link that explains most failures and adds it to the hypothesis set, until the set of links in the hypothesis set explains all path failures.

We use the following metrics to evaluate the accuracy of the hypothesis set and the speed of fault identification:

Identification rate. The percentage of failures for which the tomography algorithm finds the correct hypothesis set before the failure ends (i.e., true positives).

False alarm. A false alarm occurs when a link is working but tomography adds it to the hypothesis set.

In cases of late identification, we disregard the event: We do not consider it as a correct identification or as a false alarm. In some cases, if the reachability matrix is inconsistent, the tomography algorithm may return an empty hypothesis set. We do not consider an empty hypothesis set as a false alarm. If the failure ends and the tomography algorithm never outputs a correct hypothesis set, then we say that it missed the failure (i.e., a misidentification).

4.2 Controlled Experiments

In this section, we evaluate the benefits that failure confirmation and aggregation strategies each provide independently, using controlled experiments on Emulab. Unless otherwise stated, the setup for these experiments is as described in Sec. 3.5 with the Abilene topology with 1% per-link loss rate and average loss burst lengths of 40 ms.

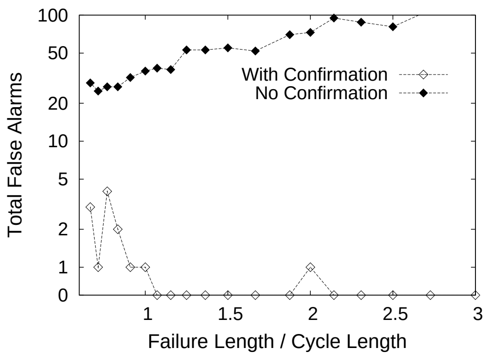


Figure 7: Effect of failure confirmation on false alarms.

4.2.1 *Effects of failure confirmation*

This section studies the accuracy of the tomography algorithm using the BASIC aggregation strategy with and without failure confirmation. We only present results on false alarm, because the confirmation mechanism has little effect on identification rate. Even with no confirmation, the tomography algorithm is still capable of identifying failures, but it will trigger many false alarms.

We configure the failure confirmation mechanism according to the guidelines presented in Sec. 2.1.3 using the loss rate and the burst length of the Emulab setup. Given the goal to achieve $F = 10^{-5}$ with $\mu_{\min} = 100$ms, we find that $\kappa = 4$ and $\mu = 398$ms.

Fig. 7 shows the absolute number of false alarms with and without confirmation when varying f/C using BASIC. Failure confirmation reduces the number of false alarms by two orders of magnitude. With no confirmation, the coordinator will run the tomography algorithm at any lost probe, and potentially trigger a false alarm. Without confirmation, the total number of false alarms increases when the cycle lengths are smaller (or when f/C increases). This increase occurs because with a smaller cycle, monitors perform more measurements during one experiment; there are more probe losses and consequently more false alarms.

Failure confirmation brings the total number of false alarms down to less than five for all values of f/C. Most false alarms occur when failures are short compared to the cycle length (as in S3 in Sec. 3.3.1), because in this scenario, BASIC produces inconsistent reachability matrices.

4.2.2 *Effects of aggregation strategies*

We now use the failure confirmation mechanism (configured as in the previous section) and compare the accuracy of the different aggregation strategies. For multi-cycle strategies, $n = 2$. Fig. 8 shows the accuracy of the tomography algorithmin experiments with 0.6% of detection errors. As usual, we vary f/C on the x-axis.

Fig. 8(a) shows the identification rate of the tomography algorithm. Similar to the analysis of consistency in Sec. 3, when failures are long (i.e., $f/C > 3$), BASIC and MC-PATH aggregation correctly identify all failures, whereas MC misses 10% of failures because of detection errors. These errors trigger path status changes, which prevent MC from obtaining a reachability matrix that is stable for n cycles. In these cases, the failure is not identified. This figure also shows

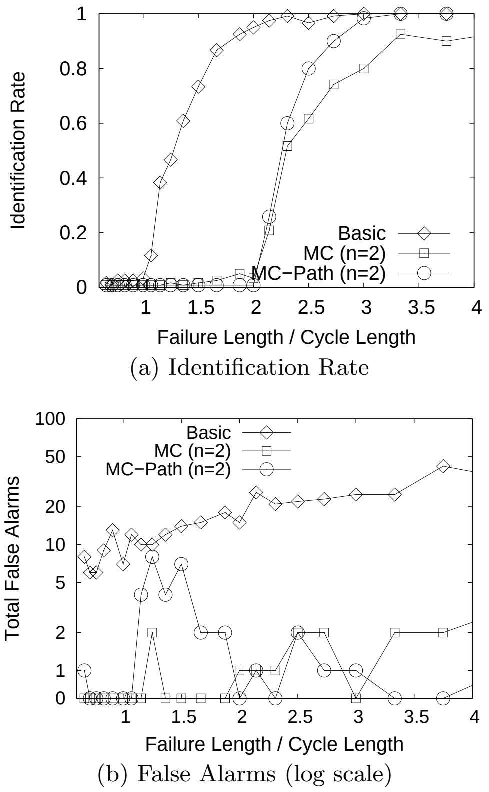


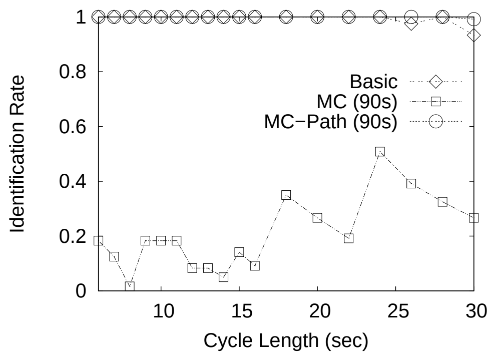
Figure 8: Effect of aggregation strategies on tomography, 0.6% of detection errors.

that BASIC cannot identify failures shorter than C. Intermediate failures (with $1 \leq f/C \leq 2$) can only be identified if BASIC builds the reachability matrix before the failure ends, i.e., only if $t_{\text{de}} \leq f - C$. Multi-cycle strategies can filter out short (i.e., $f < nC$) failures that can not be reliably identified without increasing the number of false alarms. Missing these short failures is not an issue because they do not characterize persistent blackholes.

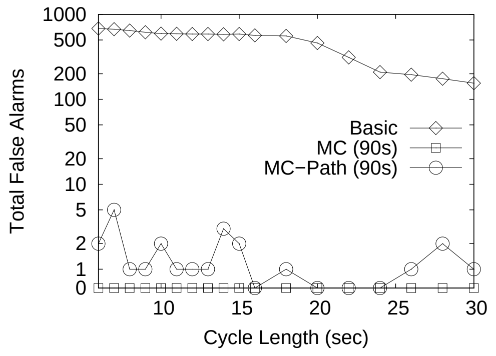
Fig. 8(b) shows the total number of false alarms triggered by the tomography algorithm using each of the aggregation strategies; the y-axis is in log scale. As discussed in Sec. 3, BASIC triggers many false alarms, whereas MC triggers the smallest number of false alarms, because it never builds a reachability matrix caused by detection errors (at the price of identifying less failures as seen in Fig. 8(a)). The number of false alarms resulting from the MC-PATH strategy does not depend significantly on detection errors.

Because we aim to identify persistent blackholes, we perform different experiments where the goal is to identify long failures without triggering false alarms due to short ones. We inject long and short failures simultaneously, varying duration of short failures from 20% to 70% of that of long failures. We explore ratios of f/C from 2, the lowest value that still guarantees consistency (Sec. 3), to 30, corresponding to very high probing rates. We configure the multi-cycle strategies to detect failures longer than a given threshold f_{target} by taking Eq. (11) into consideration and picking the largest n such that $nC < f_{\text{target}}$.

Fig. 9 presents the identification rate and number of false alarms for the experiments where longer failures last for



(a) Identification Rate



(b) False Alarms (log scale)

Figure 9: Aggregation with overlapping failures.

90 seconds and shorter ones last for 20 seconds. In contrast to the other plots in this section, the x-axis presents the cycle length. These experiments show that MC misses most of the longer failures because of the instability of path statuses created by shorter failures, which have the same effect as detection errors. All failures that MC identifies are cases where short failures only affect paths that are also affected by long failures. MC is very conservative and never triggers a false alarm. BASIC correctly identifies all long failures because there are intervals where the long failure persists but no short failure is happening, but it triggers many false alarms (Fig. 9(b)). MC-PATH is a good compromise between these extremes. It correctly identifies long failures, while keeping a small number of false alarms. All of these false alarms occur when a short failure occurs just after a long one, and this short failure affects a subset of the paths that were affected by the long failure. In this scenario, MC-PATH identifies the short failure, which we label as a false alarm.

These results confirm the findings from our analysis. Both detection errors and short failures reduce the accuracy of aggregation strategies. BASIC triggers many false alarms and MC misses some failures, so MC-PATH is a good method to use in environments where detection errors are common.

4.3 Wide-area Experiments

We apply the tomography algorithm on measurements from PlanetLab and the enterprise network to evaluate the effect of confirmation and aggregation on the number of alarms raised in a real network. We show that confirmation and multi-cycle aggregation reduces the number of alarms from thousands to a few.

	PlanetLab		Enterprise	
	NO CONF.	CONF.	NO CONF.	CONF.
BASIC	16,261	16,260	6,256	225
MC	—	—	13	17
MC-PATH	251	135	27	27

Table 3: Number of alarms in about two weeks.

We use the same measurements introduced in Sec. 3.5.2. Both experiments lasted for slightly less than two weeks. The PlanetLab experiment has 16,262 cycles of one minute ($C = 60$) and the enterprise network experiment has 228,806 cycles of five seconds each ($C = 5$). We use a conservative value for $F = 10^{-6}$, resulting in $\kappa = 7$ and $\mu = 420$ ms for PlanetLab and $\kappa = 4$ and $\mu = 280$ ms for the enterprise. We configure multi-cycle strategies to identify failures of more than 11 minutes, as a result we have $n = 10$ for PlanetLab and $n = 120$ for the enterprise network.

Tab. 3 shows the number of alarms with and without confirmation for each aggregation strategy. We have no ground truth for these deployments, so we cannot label which of these alarms are false. Considering the Emulab results from Sec. 4.2, most of the alarms we eliminate with our confirmation and aggregation strategies should be false.

These results highlight the contrast between the dynamic PlanetLab environment and the more stable enterprise network. Applying tomography to raw measurements in either network (represented by BASIC without confirmation) would lead to thousands of alarms: PlanetLab would have one alarm per minute, and the enterprise network would have one alarm every three minutes. Although the enterprise experiments have considerably more cycles, fewer than 3% of them have an alarm, whereas PlanetLab raises alarms at every cycle. Not even failure confirmation helps reduce the number of alarms in PlanetLab. This result may seem to contradict the results from Fig. 3, which shows that failure confirmation significantly reduces the fraction of confirmed path failures in PlanetLab. Failure confirmation does reduce the total number of path failures in PlanetLab from 1,739,776 to 160,777. Nevertheless, every cycle still has at least one confirmed path failure, which triggers BASIC to build a reachability matrix. Failure confirmation is more effective to remove alarms in the enterprise network.

MC-PATH's flexible aggregation strategy works for both deployments. As shown in our controlled experiments, MC is not useful in dynamic environments; it never builds a reachability matrix for PlanetLab. In the enterprise deployment, MC can detect some failures, but not as many as MC-PATH. MC-PATH with confirmation triggers 135 alarms in PlanetLab (approximately 12 alarms per day). This number is more than a hundred times smaller than using BASIC without confirmation. In the enterprise network, MC-PATH reduces the number of alarms to 2 per day. It is more realistic to expect that an operational team will deal with two alarms per day than one alarm every three minutes. We could also configure our mechanisms to only identify longer failures, which would reduce even more the number of alarms.

4.4 Comparison to State of the Art

We compare our approach to that of Kompella *et al.* [16], which is the most closely related work to ours. We present a brief overview of the previous approach, describe the experiment setup, and compare the performance of each method.

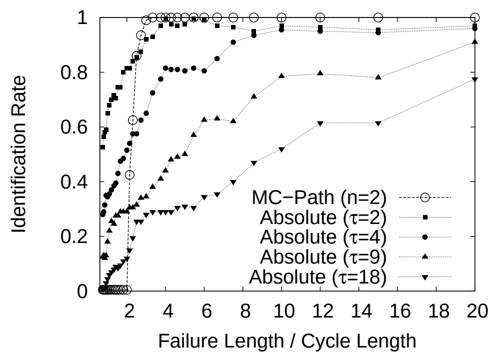


Figure 10: Comparison of identification rate.

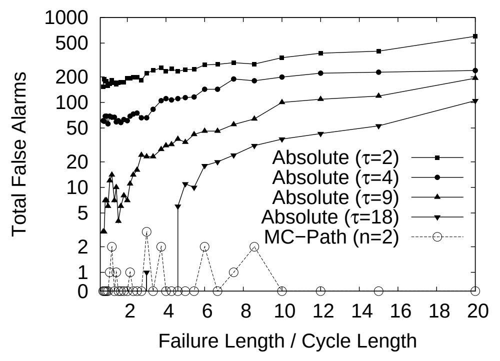


Figure 11: Comparison of number of false alarms.

We find that our approach has both a higher identification rate and lower false-alarm rate for all blackholes that last at least three cycles.

Overview and experiment setup. Kompella *et al.*'s method works as follows: (1) A coordinator groups every lost probe into a failure signature; this failure signature is a type of reachability matrix, except that each element counts the number of lost probes in a path. (2) The coordinator builds a hypothesis set for a failure signature by iteratively selecting the link that explains the largest number of lost probes. (3) The coordinator removes links from the hypothesis set that may be due to transient packet losses. In step three, we consider only their best filtering technique, called ABSOLUTE. This technique removes links from the hypothesis set if the absolute number of failures they could have caused is less than a chosen threshold, τ. We keep the network topology fixed and do not apply their techniques to merge different topology snapshots. ABSOLUTE works offline and does not use failure confirmation. Thus, we also consider our aggregation strategies without confirmation.

We compare MC-PATH and ABSOLUTE with controlled experiments in Emulab. In these experiments, we use the GEANT topology, because it is bigger than Abilene, and inject failures of 60 seconds. A bigger topology and longer failures allows for a more fair comparison; otherwise, the number of probes that traverse a failure in each measurement bin of ABSOLUTE would be too small. We vary per-link loss rates from zero to 1% and average burst lengths from 4 ms to 40 ms. We found that varying the measurement bin length in ABSOLUTE does not affect the trade-offs.

Identification rate. Fig. 10 shows the identification rate of ABSOLUTE when varying τ for different values of f/C. We show results for average path loss rates of 1.4% and average loss bursts of 40 ms (results were similar for other settings). Reducing τ allows more links in the hypothesis set, ultimately increasing the number of identified failures.

False alarms. Because ABSOLUTE only counts probe losses and does not reset these counters upon a successful probe, it raises many false alarms. Successful probes are clearly useful for removing working links from hypothesis sets, but ABSOLUTE ignores them. We favor ABSOLUTE by considering failures as correctly identified even if its hypothesis set contains some working links besides the failed links. Fig. 11 shows that decreasing τ increases the number of false alarms, because with a small threshold, ABSOLUTE will misclassify many links as failed.

For both identification rate and false alarms, no configuration of ABSOLUTE achieves the same accuracy as MC-PATH. As pointed out by the authors [16], ABSOLUTE is biased toward adding links visited by many paths to the hypothesis set, because these links tend to explain more observations in the failure signature than less visited links. Because ABSOLUTE has no failure confirmation and ignores successful probes in building failure signatures, detection errors have a high probability of causing highly visited links to be added to the hypothesis set. This approach increases false alarms and reduces identification rate if the wrong link is chosen to explain a failure. The delay (not shown) of ABSOLUTE is proportional to its bin length. The average delay of MC-PATH is lower than ABSOLUTE's for $C > 12$ (i.e., $f/C > 5$).

5. RELATED WORK

Network tomography and monitor selection. Existing binary tomography algorithms rely on end-to-end path measurements collected by monitors, as well as a coordinator that combines this information with the network topology to identify failures [9,10,16,22]. These algorithms may benefit from our confirmation and aggregation techniques. In a recent survey, Huang *et al.* stated the challenges of using binary tomography in practice [14]. The methods we propose in this paper systematically address these challenges; our methods are more flexible and adaptable to various network conditions and achieve higher identification rates with fewer false alarms. Other previous work has developed path and monitor selection algorithms [2, 19, 30], which reduce the number of paths to probe and, consequently, the cycle length. Our analysis formally relates the cycle length to overall aggregation delay for various aggregation strategies.

Fault identification systems. NICE [18] correlates different data sources from an ISP network to identify intermittent failures. The method incorporates end-to-end measurements but focuses mostly on failures that already appear in the network's alarm system (as opposed to blackholes). The alarms from a system based on binary tomography could be one more data source for NICE. PlanetSeer [28] passively monitors TCP traffic in PlanetLab nodes to detect paths experiencing problems and triggers traceroutes from a set of vantage points to the effected destinations. Hubble [15] uses RouteViews data and low-rate pings to run traceroutes to destinations that are more likely to be experiencing reachability problems. Both systems use traceroutes to identify problems in the Internet, but traceroutes often reveal the effect of a failure (i.e., where paths stop), but not necessarily its location. It is not known to what extent PlanetSeer and Hubble are accurate, since both were evaluated in the wide-area Internet without a notion of ground truth; in contrast, we have evaluated our methods in a controlled setting to determine the accuracy and false alarm rates of our methods for various failure scenarios.

6. CONCLUSION

Binary tomography algorithms hold great promise for helping network operators detect and locate a large class of network failures, including those that are difficult to detect with other methods (e.g., network "blackholes"). Despite much previous attention to binary tomography algorithms, two significant obstacles have prevented network tomography from being applied in practice: (1) the inability to distinguish persistent blackholes from bursty, congestion-related losses; and (2) the lack of synchronized end-to-end measurements. This paper has designed and evaluated *confirmation* and *aggregation* methods that address these two problems, respectively. We generated analytical models to explain the accuracy of detection and consistency of the reachability matrix, validated these models using controlled experiments on the Emulab testbed, and showed that these methods quickly and accurately identify all failures that are longer than two measurement cycles, with few false alarms. In our future work, we plan to use these methods in conjunction with the algorithms themselves to build a real-time, tomography-based monitoring system that can detect and locate network blackholes in real time.

Acknowledgements

We thank Darryl Veitch and Laurent Massoulié for their feedback on our failure confirmation mechanism. This work was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) no. 223850, the ANR project C'Mon, NSF CNS-0721581, and NSF CAREER Award 0643974. Nick Feamster was also partially supported by Thomson.

7. REFERENCES

[1] F. Baccelli, S. Machiraju, D. Veitch, and J. Bolot. The Role of PASTA in Network Measurement. *SIGCOMM CCR*, 36(4):231–242, 2006.

[2] P. Barford, N. Duffield, A. Ron, and J. Sommers. Network Performance Anomaly Detection and Localization. In *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, 2009.

[3] P. Barford and J. Sommers. Comparing Probe- and Router-Based Packet-Loss Measurement. *Internet Computing*, 8(5):50–56, 2004.

[4] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, and G. Salmon. Experimental Study of Router Buffer Sizing. In *Proc. IMC*, Vouliagmeni, Greece, 2008.

[5] J. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive FEC-Based Error Control for Internet Telephony. In *Proc. IEEE INFOCOM*, New York, NY, 1999.

[6] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley. Multicast-based Inference of Network-internal Loss Characteristics. *IEEE Trans. on Information Theory*, 45(7):2462 – 2480, 1999.

[7] M. Caesar and J. Rexford. Building Bug-tolerant Routers with Virtualization. In *Proc. SIGCOMM PRESTO*, Seattle, WA, 2008.

[8] I. Cunha, R. Teixeira, N. Feamster, and C. Diot. Fast and Accurate Blackhole Identification with Network Tomography. *Thomson Technical Report CR-PRL-2009-05-0006*.

[9] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot. NetDiagnoser: Troubleshooting Network Unreachabilities Using End-to-end Probes and Routing Data. In *Proc. CoNEXT*, New York, NY, 2007.

[10] N. G. Duffield. Network Tomography of Binary Network Performance Characteristics. *IEEE Trans. on Information Theory*, 52(12):5373–5388, 2006.

[11] L. Fang, A. Atlas, F. Chiussi, K. Kompella, and G. Swallow. LDP Failure Detection and Recovery. *IEEE Communication Magazine*, 42(10):117–123, 2004.

[12] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *Proc. NSDI*, Boston, MA, 2005.

[13] E. Gilbert. Capacity of a Burst-Noise Channel. *Bell Systems Technical Journal*, 39(5):1253–1265, 1960.

[14] Y. Huang, N. Feamster, and R. Teixeira. Practical Issues with Using Network Tomography for Fault Diagnosis. *SIGCOMM CCR*, 38(5):53–58, 2008.

[15] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. Anderson. Studying Black Holes in the Internet with Hubble. In *Proc. NSDI*, San Francisco, CA, 2008.

[16] R. Kompella, J. Yates, A. Greenberg, and A. Snoeren. Detection and Localization of Network Blackholes. In *Proc. IEEE INFOCOM*, Anchorage, AK, 2007.

[17] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP Misconfiguration. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, 2002.

[18] A. Mahimkar, J. Yates, Y. Zhang, A. Shaikh, J. Wang, Z. Ge, and C. Ee. Troubleshooting Chronic Conditions in Large IP Networks. In *CoNEXT*, Madrid, Spain, 2008.

[19] H. Nguyen and P. Thiran. Active Measurement for Multiple Link Failure Diagnosis in IP Networks. In *Proc. PAM*, Antibes-Juan les Pins, France, 2004.

[20] V. Padmanabhan, L. Qiu, and H. Wang. Server-based Inference of Internet Link Lossiness. In *Proc. IEEE INFOCOM*, San Francisco, CA, 2003.

[21] K. Papagiannaki, R. Cruz, and C. Diot. Network Performance Monitoring at Small Time Scales. In *Proc. IMC*, Miami Beach, FL, 2003.

[22] M. Rabbat, M. Coates, and R. Nowak. Multiple Source, Multiple Destination Network Tomography. In *Proc. IEEE INFOCOM*, Hong Kong, China, 2004.

[23] R. Sherwood, A. Bender, and N. Spring. DisCarte: a Disjunctive Internet Cartographer. In *Proc. ACM SIGCOMM*, Seattle, WA, 2008.

[24] J. Sommers and P. Barford. An Active Measurement System for Shared Environments. In *Proc. IMC*, San Diego, CA, 2007.

[25] J. Sommers, P. Barford, N. Duffield, and A. Ron. Improving Accuracy in End-to-end Packet Loss Measurement. In *Proc. ACM SIGCOMM*, Philadelphia, PA, 2005.

[26] J. Sommers, P. Barford, N. Duffield, and A. Ron. Accurate and Efficient SLA Compliance Monitoring. In *Proc. ACM SIGCOMM*, Kyoto, Japan, 2007.

[27] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, 2002.

[28] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-area Services. In *Proc. OSDI*, San Francisco, CA, 2004.

[29] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Consistency of Internet Path Properties. In *Proc. IMW*, San Francisco, CA, 2001.

[30] Y. Zhao, Z. Zhu, Y. Chen, D. Pei, and J. Wang. Towards Efficient Large-Scale VPN Monitoring and Diagnosis under Operational Constraints. In *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, 2009.