Feedback Verification for Trustworthy Tomography

Vijay Arya, Thierry Turletti, Ceilidh Hoffmann

INRIA, Sophia Antipolis, France

E-mail: {`vijay.arya, thierry.turletti, ceilidh.hoffmann`}`@sophia.inria.fr`

Abstract

Network tomography is a process by which internal characteristics of a network are inferred from "external" end-to-end measurements. To ensure that the inferred internal characteristics are sound and trustworthy, it is essential to verify the integrity of data collected from external measurements. In this paper, we present an algorithm which can verify the integrity of data collected from end-to-end multicast measurements. This data is used by a multicast-based tomography tool called MINC to infer loss rates on internal network links. MINC performs loss inference by analyzing binary feedbacks reported by receivers in response to multicast probes sent from the source. However, buggy or malicious receivers can report incorrect feedbacks, resulting in a faulty loss inference. In this work, we consider the problem of verifying the integrity of binary feedbacks collected from receivers of a multicast tree to ensure a sound and trustworthy MINC loss inference. We start by showing how the MINC loss inference becomes erroneous if feedbacks of receivers are altered. Then, we present a statistical verification algorithm which checks if feedbacks of receivers are consistent with respect to one another. We present the performance of this algorithm on Model-based traces, NS traces and MBone loss traces.

1 Introduction

Network tomography refers to the process of inferring internal characteristics of a network from end-to-end measurements. *Multicast-based Inference of Network internal Characteristics* (MINC) is one of the earliest proposed methods of performing network tomography [1,2,9]. MINC can infer internal characteristics of a network which lies under a multicast tree using end-to-end multicast measurements. MINC can infer characteristics such as loss rates and delay distributions of network links [2, 9]. To infer loss rates, the source sends a stream of probe packets into the multicast tree. For each probe, each receiver reports whether it received the probe (`1`) or not (`0`). Based on the binary feedback traces collected from all receivers, per link loss rates in the multicast tree are inferred. The loss rate of a link indicates the level of congestion on that link. MINC loss inference is thus useful in monitoring the level of congestion on specific network links. It can be used by network operators and service providers to identify congested links in the underlying network and manage their traffic efficiently or upgrade certain parts of their network. The RTCP [5] format also has been extended to perform MINC measurements [3]. Multicast sessions which use RTP [12] can use their data packets as probes and RTCP to report feedbacks. When used in this manner with RTCP, MINC loss inference can also be utilized by multicast congestion control protocols to detect bottleneck links in the multicast tree.

Relying *solely* on end-to-end receiver feedbacks to infer internal network characteristics is the essence of tomography. However, in order to utilize an inference which is based on "external" end-to-end measurements to make important decisions about the network, it is essential to ensure that the external data is indeed correct. One of the causes of erroneous inference is incorrect feedbacks from buggy receivers. Due to configuration errors, software patches, and several multi-platform implementations, often networking software is buggy and does not function as intended [4,15]. For instance, in [8], bugs found during experimentation with NIMI have been reported. RFC 2525 lists 18 common bugs found in TCP implementations. Designers of NetLogger [13] have pointed that 45% of problems in distributed applications arise due to the presence of bugs in networking software. Thus, to ensure a sound and trustworthy inference, it becomes necessary to verify the integrity of measured feedback data.

Further, since decisions made by utilizing network inference eventually effect network users, this gives malicious users an incentive to report incorrect feedbacks and obtain a favorable inference. In multicast congestion control, a receiver can report wrong feedbacks and mislead the congestion control protocol to increase its sending rate, thereby harming other well behaved flows in the network [6]. In sender-based multicast congestion control schemes, the slowest receiver can report wrong feedbacks and cause the source to inflate its sending rate, to obtain a better bandwidth. Loss rate snapshots inferred by MINC can be used to find out if there is persistent congestion in the multicast tree when the sending rate increases. However, misbehaving receivers can also report wrong feedbacks to MINC to hide their congestion related misbehavior. In such a setting, before using the binary feedbacks from multicast recievers for MINC inference, they need to be checked for correctness.

This work presents an integrity check which verifies binary feedbacks collected from multicast receivers in order to ensure a sound and trustworthy loss inference. Due to inherent correlations in multicast traffic, loss rates of paths in the multicast tree can be inferred in different ways. Our work exploits this idea to design a statistical verification procedure which detects loss rate inconsistencies that arise in erroneous feedback data. Furthermore, in conformance with the end-to-end nature of tomography, our procedure does not require any knowledge of the multicast tree topology.

1.1 Contributions

We present two related contributions in this paper. Our first contribution is an analysis which explains how the loss rates inferred by MINC change when receivers report incorrect feedbacks. Our analysis shows that when receivers falsely report that they received the probe packet, the loss rates inferred by MINC in a large portion of the multicast tree can get altered.

For MINC loss inference, the binary feedbacks for N probes from R receivers are available in the form of a $N \times R$ binary feedback matrix. Given such a matrix that potentially contains incorrect feedbacks, the following questions can be posed: *(a) Is the given data erroneous, or equivalently, are the feedbacks of one or more receivers incorrect? (b) Which are the receivers whose feedbacks are incorrect? (c) In spite of errors, can we make the right MINC loss inference using the given feedback data?*

Our main contribution is an algorithm called *ICheck* which answers (a). *ICheck* is a statistical procedure which searches for loss rate inconsistencies that arise in erroneous feedback data. Broadly speaking, *ICheck* conducts statistical tests to determine the likelihood of collecting the given feedback data from receivers of a multicast tree. *ICheck* uses the core principle of MINC loss inference itself. Like MINC, it is based on the premise that a probe is lost on a link using Bernoulli loss process, i.e., successive probe packets are lost independently (which is true in the presence of sufficient background Internet traffic [1]). *ICheck* takes as input only the $N \times R$ binary matrix and does not require any knowledge of the multicast tree topology.

The rest of the paper is organized as follows. Section 2 briefly explains how loss rates are inferred in MINC. Section 3 examines how the loss rates inferred by MINC change due to incorrect feedbacks. Section 4 presents the *ICheck* algorithm for feedback verification. Section 5 presents experimental results. Sections 6 and 7 present discussions and conclusions respectively.

2 MINC

In this section, the principle used by MINC to infer the loss rates or the passage rates of links is described; passage rate = 1 $-$ loss rate (The terms loss rate and loss probability are equivalent and so are passage rate and passage probability. If out of n packets sent on a link, m are lost, then the loss probability of the link is m/n and its passage probability is $(n-m)/n$). MINC infers the characteristics

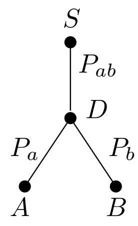


Figure 1. Multicast tree with two receivers

of a network underlying a multicast tree by exploiting the inherent correlation in multicast traffic. MINC infers loss rates in the logical multicast tree. Each link in the logical tree is a series of physical links in the underlying network between two branch points. Consider the logical multicast tree shown in figure 1 with source S, two receivers A, B and the branch node D. Suppose that the source sends a stream of probe packets and each receiver observes whether it received the probe (`1`) or not (`0`). Consider the task of estimating the passage probability of the path DB. For this, consider those packets which were received by A. Since A received them, these multicast packets must have crossed the branch node D and also sent out on path DB. *Among them*, some may have crossed the path DB and some lost on this path. Thus, the ratio of the number of packets which both A and B received to the total number of packets which A received estimates the passage probability of path DB. The passage probability of path DA can be calculated similarly.

Formally, suppose that the sender injects N probe packets into the multicast tree. Let (i, j), $i, j \in \{0, 1\}$ denote the *probe* corresponding to which A reported i and B reported j. Equivalently, (i, j) also denotes the feedback itself where A reported i and B reported j. Let n_{ij} denote the total number of probes of type (i, j). For example, n_{10} denotes the total number of probes for which A reported 1 and B reported 0. We extend this notation slightly by allowing $i, j \in \{0, 1, *\}$, where "$*$" means a *dont care* (either a 0 or 1). For example, n_{1*} denotes the total number of probes for which A reported 1 and B reported either a 0 or 1; $n_{1*} = n_{10} + n_{11}$. Now, the passage probability of the path DB, denoted by P_b and the passage probability of path DA, denoted by P_a are given by

$$P_b = \frac{n_{11}}{n_{1*}}, \;\; P_a = \frac{n_{11}}{n_{*1}} \tag{1}$$

Similarly, the loss probabilities of path DB and DA are

$$\bar{P}_b = \frac{n_{10}}{n_{1*}}, \;\; \bar{P}_a = \frac{n_{01}}{n_{*1}} \tag{2}$$

Having done this, the passage probability of path SD denoted by P_{ab} can be estimated as follows:

$$P_{ab} = \frac{n_{11}/n}{P_a \cdot P_b} = \frac{n_{*1} n_{1*}}{N \cdot n_{11}} \tag{3}$$

Since SD is the common path between A and B, P_{ab} is also called the *common passage probability*. The above principle is extended in MINC to calculate the passage probabilities of all the paths in the multicast tree. To perform MINC loss inference, the topology of the multicast tree is needed. However, the common passage probabilities calculated between different receiver pairs can also be utilized to infer the topology of the multicast tree [1, 11]. Thus, MINC loss tomography can be performed entirely in an end-to-end manner.

2.1 Simple Observations I

An observation which aids the subsequent analysis is now made. Consider the (00) probe. This probe was either *(i)* lost on the path SD (denoted *common loss*) or *(ii)* it crossed SD and was lost simultaneously on paths DA and DB (denoted *independent loss*). We classify the (00) probes into these two respective categories. Let n^c_{00} denote the total number of probes lost on path SD. Let n^i_{00} denote the total number of probes which crossed SD and were lost simultaneously on DA and DB. Now, it is noted that $n^i_{00} + n_{01}$ are the total number of probes which crossed the path SD and lost on DA. *Among them,* n_{01} crossed DB and n^i_{00} were lost on DB. Thus, the passage probability P_b can also be written as

$$P_b = \frac{n_{01}}{n^i_{00} + n_{01}} \tag{4}$$

Thus, from (1) and (4) we have

$$P_b = \frac{n_{01}}{n^i_{00} + n_{01}} = \frac{n_{11}}{n_{1*}} \tag{5}$$

3 Misbehavior and its impact on passage probabilities

We use the term "misbehaving receiver" to denote the buggy or malicious receivers which can report incorrect feedbacks. A misbehaving receiver can misbehave either by altering a feedback from 0 to 1 (denoted by $0 \rightsquigarrow 1$) or by altering a feedback from 1 to 0 (denoted by $1 \rightsquigarrow 0$). When a receiver misbehaves from $0 \rightsquigarrow 1$, it reports that it received the probe when it actually did not. When it misbehaves from $1 \rightsquigarrow 0$, it reports that it did not receive the probe when it actually did. If a receiver reports a wrong feedback, the passage probabilities inferred by MINC in the multicast tree change. Figure 2(a) shows the impact of misbehavior on the passage probabilities inferred by MINC when receiver A alters some of its feedbacks from $0 \rightsquigarrow 1$. The passage probability of the path from A to the source increases ($\Uparrow$) and the passage probabilities of all links connected to this path decrease ($\Downarrow$). Thus passage probabilities in a large region of the multicast tree are altered. Figure 2(b) shows the impact of misbehavior on the passage probabilities inferred by MINC when receiver A alters some of its feedbacks from $1 \rightsquigarrow 0$. In this case only the passage probability of the path from A to its parent decreases($\Downarrow$) to be congruous with the data reported by A. The passage probabilities in the rest of the multicast tree remain unchanged.

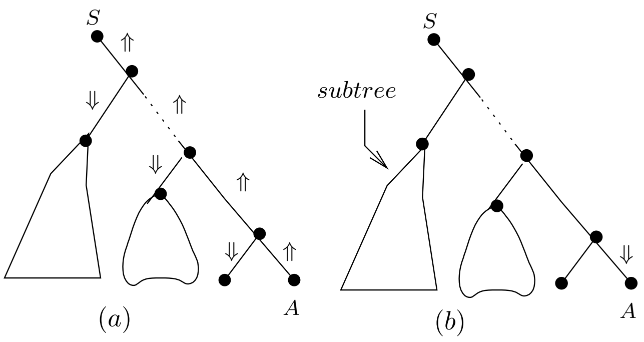


Figure 2. Effects of misbehavior on passage probabilities. (a) Receiver A reports more 1's, (b) Receiver A reports more 0's

To explain the above changes in probabilities, the misbehavior mechanism of a user is modeled as follows. It is assumed that a receiver j misbehaves with probability α_j ($0 \leq \alpha_j \leq 1$). If receiver j misbehaves from $0 \rightsquigarrow 1$, it changes its 0 feedback to 1 with probability α_j and vice versa.

Consider the two receiver system shown in figure 1 with receivers A, B and sender S. Now, we calculate the *expected* passage or loss probabilities after misbehavior. Firstly, the misbehavior from $0 \rightsquigarrow 1$ is considered.

3.1 Receiver A misbehaves from $0 \rightsquigarrow 1$

After A misbehaves, let $\mathbb{P}_a$, $\mathbb{P}_b$ and $\mathbb{P}_{ab}$ denote the respective altered passage probabilities.

Lemma 1. *If A misbehaves from $0 \rightsquigarrow 1$ with probability α_a, (i) $E[\mathbb{P}_b] \le P_b$ (ii) $E[\mathbb{P}_a] \ge P_a$ (iii) $E[\mathbb{P}_{ab}] \ge P_{ab}$*

Proof. When A misbehaves from $0 \rightsquigarrow 1$, it causes two types of probe or feedback transformations:

$$x : (00) \Rightarrow (10) \;\; y : (01) \Rightarrow (11)$$

The following table shows the *expected* feedback system after these transformations.

Original		After Misbehavior
n_{00}		$n_{00}(1-\alpha_a)$
n_{01}	$\Rightarrow$	$n_{01}(1-\alpha_a)$
n_{10}		$n_{10}+\alpha_a \cdot n_{00}$
n_{11}		$n_{11}+\alpha_a \cdot n_{01}$

The altered passage probability of path DB denoted by $\mathbb{P}_b$ will be,

$$\begin{aligned} E[\mathbb{P}_b] &= \frac{n_{11}+\alpha_a \cdot n_{01}}{n_{1*}+\alpha_a \cdot n_{00}+\alpha_a \cdot n_{01}} \\ &= \frac{n_{11}+\alpha_a \cdot n_{01}}{n_{1*}+\alpha_a(n^i_{00}+n_{01})+\alpha_a \cdot n^c_{00}} \end{aligned} \tag{6}$$

We know that $a/b = c/d \Rightarrow (a+c)/(b+d) = a/b = c/d$. Applying this to equation (5), we get

$$P_b = \frac{n_{11}+\alpha_a \cdot n_{01}}{n_{1*}+\alpha_a(n^i_{00}+n_{01})} \tag{7}$$

Subtracting (6) from (7) gives,

$$E[\mathbb{P}_b] = P_b \left\{ 1 - \frac{n^c_{00} \cdot \alpha_a}{n_{1*}+n_{0*} \cdot \alpha_a} \right\} \qquad \Downarrow \tag{8}$$

Now,

$$E[\bar{\mathbb{P}}_a] = \frac{n_{01}(1-\alpha_a)}{n_{*1}} = \bar{P}_a(1-\alpha_a) \qquad \Downarrow \tag{9}$$

Proof for *(iii)* is similar to that of *(i)*. □

Intuitively, when A's feedbacks are altered from $0 \rightsquigarrow 1$, those feedbacks for which both A and B had reported 0, i.e. of the form (00), change to (10). Some of these (00) feedbacks correspond to probes which were lost on the path SD. These probes are now counted as having crossed SD and lost on DB, causing the passage probability of SD to increase and the passage probability of DB to decrease. In general, suppose that A misbehaves by causing n_x transformations of type x and n_y transformations of type y. The following corollary gives the conditions for observing the expected changes in probabilities after misbehavior.

Corollary 1. *If receiver A misbehaves from $0 \rightsquigarrow 1$ resulting in n_x and n_y transformations, then $\mathbb{P}_b < P_b$ and $\mathbb{P}_{ab} > P_{ab}$ if $n_x/n_y > n_{10}/n_{11}$.*

Proof. After misbehavior we have,

$$\mathbb{P}_b = \frac{n_{11}+n_y}{n_{1*}+n_x+n_y}$$

$$\mathbb{P}_b < P_b \text{ iff } \quad \frac{n_{11}+n_y}{n_{1*}+n_x+n_y} < \frac{n_{11}}{n_{1*}}$$

$$\text{i.e., iff} \quad (n_{1*}-n_{11})n_y < n_{11}n_x$$

$$\text{i.e., iff} \quad \frac{n_x}{n_y} > \frac{n_{10}}{n_{11}} = \frac{\bar{P}_a}{P_a}$$

Proof for P_{ab} is similar. □

Corollary 2. *If both receivers A and B misbehave from $0 \rightsquigarrow 1$ with the same probability, the receiver which suffers more losses before misbehavior causes a greater decrease in the passage probability of the other receiver.*

Proof. After A and B misbehave from $0 \rightsquigarrow 1$ with probabilities α_a and α_b, the expected probe system is shown below.

Original		After Misbehavior
n_{00}		$n_{00}(1-\alpha_a)(1-\alpha_b)$
n_{01}	$\Rightarrow$	$n_{01}(1-\alpha_a)+\alpha_b(1-\alpha_a)n_{00}$
n_{10}		$(n_{10}+\alpha_a \cdot n_{00})(1-\alpha_b)$
n_{11}		$n_{11}+\alpha_a \cdot n_{01}+\alpha_b(n_{10}+\alpha_a \cdot n_{00})$

Working out $E[\bar{\mathbb{P}}_a]$ and $E[\bar{\mathbb{P}}_b]$ as before, we have

$$E[\bar{\mathbb{P}}_a] = (1-\alpha_a)\left\{ \frac{n_{01}+\alpha_b \cdot n_{00}}{n_{*1}+\alpha_b \cdot n_{*0}} \right\} \tag{10}$$

$$E[\bar{\mathbb{P}}_b] = (1-\alpha_b)\left\{ \frac{n_{10}+\alpha_a \cdot n_{00}}{n_{1*}+\alpha_a \cdot n_{0*}} \right\} \tag{11}$$

If $\alpha_a = \alpha_b$, the increase in each of the above depends on the ratios n_{00}/n_{*0} and n_{00}/n_{0*} respectively. □

3.2 Receiver A misbehaves from $1 \rightsquigarrow 0$

After A misbehaves, let $\mathbb{P}_a$, $\mathbb{P}_b$, and $\mathbb{P}_{ab}$ denote the respective altered passage probabilities.

Lemma 2. *If A misbehaves from $1 \rightsquigarrow 0$ with probability α_a, (i) $E[\mathbb{P}_b] = P_b$ (ii) $E[\mathbb{P}_a] \le P_a$ (iii) $E[\mathbb{P}_{ab}] = P_{ab}$*

Proof. If a receiver misbehaves from $1 \rightsquigarrow 0$, it causes two types of probe transformations: $\bar{x} : (10) \Rightarrow (00)$ and $\bar{y} : (11) \Rightarrow (01)$. Writing down the expected probe system after transformations as before, we will have

Original		After Misbehavior
n_{00}		$n_{00}+\alpha_a \cdot n_{10}$
n_{01}		$n_{01}+\alpha_a \cdot n_{11}$
n_{10}	$\Rightarrow$	$n_{10}(1-\alpha_a)$
n_{11}		$n_{11}(1-\alpha_a)$

$$E[\mathbb{P}_b] = \frac{n_{11}(1-\alpha_a)}{n_{1*}(1-\alpha_a)} = P_b$$
$$E[\mathbb{P}_a] = \frac{n_{11}(1-\alpha_a)}{n_{*1}} = P_a(1-\alpha_a) \quad \Downarrow$$

Proof for *(iii)* is same as that for *(i)* □

In general, suppose that A misbehaves by causing $n_{\bar{x}}$ and $n_{\bar{y}}$ transformations. The following corollary gives the general conditions for observing the expected changes in probabilities after misbehavior.

Corollary 3. *If A misbehaves from $1 \rightsquigarrow 0$ resulting in $n_{\bar{x}}$ and $n_{\bar{y}}$ transformations, then $\mathbb{P}_b = P_b$, $\mathbb{P}_{ab} = P_{ab}$ if $n_{\bar{x}}/n_{\bar{y}} = n_{10}/n_{11}$.*

Proof. After misbehavior we have,

$$\mathbb{P}_b = \frac{n_{11} - n_{\bar{y}}}{n_{1*} - n_{\bar{x}} - n_{\bar{y}}}$$

$$\mathbb{P}_b = P_b \ \text{ iff } \ \frac{n_{11} - n_{\bar{y}}}{n_{1*} - n_{\bar{x}} - n_{\bar{y}}} = \frac{n_{11}}{n_{1*}}$$

$$ie., \ \text{ iff } \ (n_{1*} - n_{11})n_{\bar{y}} = n_{11}n_{\bar{x}}$$

$$ie., \ \text{ iff } \ \frac{n_{\bar{x}}}{n_{\bar{y}}} = \frac{n_{10}}{n_{11}} = \frac{\bar{P}_a}{P_a}$$

Similarly for $\mathbb{P}_{ab}$. □

The rest of the paper concentrates on $0 \rightsquigarrow 1$ misbehavior, since $1 \rightsquigarrow 0$ misbehavior has almost no impact.

4 Algorithm for feedback verification

Broadly, given the $N \times R$ binary feedback matrix, the *ICheck* algorithm considers three receivers at a time and verifies the feedbacks of two particular receivers using the feedbacks of the third receiver. This verification is done by performing a test on the feedbacks of the three receivers. In this section, we describe this test in Lemma 3. Subsequently, we describe the *ICheck* algorithm.

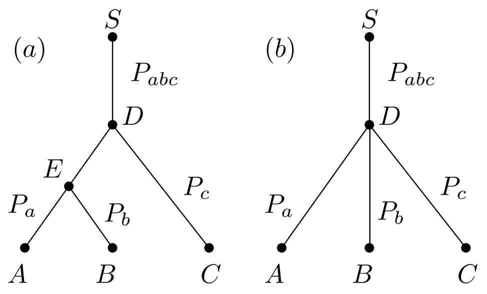


Figure 3. Multicast trees with three receivers

4.1 Simple Observations II

Figure 3 shows the two possible topologies which connect an arbitrary set of three receivers within a multicast tree. Consider the topology of Fig 3(a) with receivers A, B, C, and sender S. With usual notation, let n_{ijk} denote the number of probes for which A reports i, B reports j, and C reports k. Let P_c, P_a, P_b, and P_{abc} denote the passage probabilities of paths DC, EA, EB, and SD respectively. Consider the problem of estimating the passage probability of path DC. Observe that, if either A or B received a probe, this probe must have reached the point D and must have been sent out on the path DC. Now, n_{10*} is a sample of probes received by A. Thus, these probes crossed SD and were sent out on DC. Among them, n_{100} were lost on DC and n_{101} crossed DC. Thus P_c can be estimated as

$$P_c = \frac{n_{101}}{n_{10*}} \tag{12}$$

Using the same reasoning for the sample of probes n_{01*} which were received by B, P_c can also be estimated as

$$P_c = \frac{n_{011}}{n_{01*}} \tag{13}$$

From (12) and (13), we have

$$\frac{n_{100}}{n_{101}} = \frac{n_{010}}{n_{011}} = \frac{\bar{P}_c}{P_c} \tag{14}$$

Lemma 3 will show that the two ratios in (14) differ when feedbacks of either or both A and B are altered. An observation similar to the one in section 2.1 is now made. The n_{000} probes are split into two groups - n^c_{000} and n^i_{000}. Now, n^c_{000} are the number of probes lost on the path SD. n^i_{000} are those which crossed SD and were lost simultaneously in the *left subtree* rooted at D and on the path DC. Now it is observed that $(n^i_{000} + n_{001})$ crossed SD and were lost in the left subtree. Among them, n^i_{000} were lost on DC and n_{001} crossed DC. Thus we have,

$$\frac{n^i_{000}}{n_{001}} = \frac{\bar{P}_c}{P_c} \tag{15}$$

Lemma 3. *After A and B misbehave from $0 \rightsquigarrow 1$ with probabilities α_a and α_b, in the expected new system*

$$\frac{\mathbbm{n}_{100}}{\mathbbm{n}_{101}} \neq \frac{\mathbbm{n}_{010}}{\mathbbm{n}_{011}}$$

except when either of these conditions hold

(i) $P_{abc} = 1$

(ii) $\alpha_a/\alpha_b = n_{101}/n_{011}$

Proof. After A and B misbehave, the relevant part of the expected new system is shown below. $\mathbbm{n}_{ijk}$ denotes n_{ijk} in the expected system.

$$
\begin{aligned}
\mathbb{n}_{010} &= (n_{010} + \alpha_b \cdot n_{000})(1-\alpha_a)\\
\mathbb{n}_{011} &= (n_{011} + \alpha_b \cdot n_{001})(1-\alpha_a)\\
\mathbb{n}_{100} &= (n_{100} + \alpha_a \cdot n_{000})(1-\alpha_b)\\
\mathbb{n}_{101} &= (n_{101} + \alpha_a \cdot n_{001})(1-\alpha_b)
\end{aligned}
$$

Thus,

$$
\begin{aligned}
\frac{\mathbb{n}_{100}}{\mathbb{n}_{101}} &= \frac{n_{100} + \alpha_a \cdot n_{000}}{n_{101} + \alpha_a \cdot n_{001}}\\
&= \frac{n_{100} + \alpha_a \cdot n^i_{000}}{n_{101} + \alpha_a \cdot n_{001}} + \frac{\alpha_a \cdot n^c_{000}}{n_{101} + \alpha_a \cdot n_{001}} \quad (16)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\mathbb{n}_{010}}{\mathbb{n}_{011}} &= \frac{n_{010} + \alpha_b \cdot n_{000}}{n_{011} + \alpha_b \cdot n_{001}}\\
&= \frac{n_{010} + \alpha_b \cdot n^i_{000}}{n_{011} + \alpha_b \cdot n_{001}} + \frac{\alpha_b \cdot n^c_{000}}{n_{011} + \alpha_b \cdot n_{001}} \quad (17)
\end{aligned}
$$

From equations (14) and (15) we have,

$$
\frac{n_{100} + \alpha_a \cdot n^i_{000}}{n_{101} + \alpha_a \cdot n_{001}} = \frac{n_{010} + \alpha_b \cdot n^i_{000}}{n_{011} + \alpha_b \cdot n_{001}}
$$

Thus, $\mathbb{n}_{100}/\mathbb{n}_{101} = \mathbb{n}_{010}/\mathbb{n}_{011}$ if

$$
\frac{\alpha_a \cdot n^c_{000}}{n_{101} + \alpha_a \cdot n_{001}} = \frac{\alpha_b \cdot n^c_{000}}{n_{011} + \alpha_b \cdot n_{001}} \tag{18}
$$

i.e., iff

(i) $n^c_{000} = 0 \Rightarrow P_{abc} = 1$ or

(ii) $\alpha_a/\alpha_b = n_{101}/n_{011}$

□

Condition (ii) above essentially implies that

$$
\frac{\alpha_a}{\alpha_b} = \frac{P_a(1-P_b)}{P_b(1-P_a)} \tag{19}
$$

If A and B misbehave with the same probability then condition (19) *does not* hold *unless* P_a is also equal to P_b. Thus, even if A and B misbehave with the same probability, the two ratios n_{100}/n_{101} and n_{010}/n_{011} would differ in most scenarios. Also, P_{abc} determines the amount of n^c_{000} probes available to distort the two ratios, making them different after misbehavior. If P_{abc} is low, there is a higher chance that the two ratios would differ more after misbehavior.

Lemma 4. *If C misbehaves from $0 \rightsquigarrow 1$, in the expected new system, both estimates of P_c (12) and (13) remain equal.*

Proof. Assuming that C misbehaves with probability α_c, after misbehavior we have,

$$
E[\bar{\mathbb{P}}_c] = \frac{n_{010}(1-\alpha_c)}{n_{01*}} = \frac{n_{100}(1-\alpha_c)}{n_{10*}}
$$

□

As a result of lemma 4, we have that lemma 3 holds irrespective of whether C misbehaves or not. (It is now noted that the above results of lemma 3 and 4 hold for the other 3-receiver topology of Fig 3(b) as well).

4.2 ICheck

```
Procedure ICheck(F, k, δ)
F[N × R] : Binary feedback matrix
k ≤ (R choose 3) : Times to repeat
δ : confidence level
 1: inconsistent ← 0
 2: while k > 0 do
 3:     (x, y, z) = Random(N) //same set not repeated
 4:     (A, B, C) = LabelTree(F, x, y, z)
 5:     failed ← HTest(F[A], F[B], F[C], δ)
 6:     if failed then
 7:         inconsistent ← inconsistent + 1
 8:     end if
 9:     k ← k − 1
10: end while
11: print inconsistent
```

```
Procedure LabelTree(F, x, y, z)
 1: Compute P_xy, P_yz, P_zx
 2: temp ← min(P_xy, P_yz, P_zx)
 3: if temp = P_xy then
 4:     return(x, y, z)
 5: else if temp = P_yz then
 6:     return(y, z, x)
 7: else
 8:     return(z, x, y)
 9: end if
```

Figure 4. *ICheck* **Algorithm**

Figure 4 presents the algorithm for feedback verification. *ICheck* examines the entire feedback data by considering feedbacks of three random receivers each time and applying the test of lemma 3 to detect inconsistencies. The test of Lemma 3 is applied in *HTest*. Lemma 3 tests the feedbacks of receivers A and B using the feedbacks of C. To apply this test on an arbitrary set of 3 receivers, *ICheck* needs to identify which of these receivers can function as C. For this, it uses the *LabelTree* procedure (Fig 4). This procedure uses a principle from [1, 11] and labels the pair of receivers with minimum common passage probability as (A, B) and the other as C (Fig 5). Thus probes received by A or B would have also been sent out on the link DC. However, due to excessive feedback alterations, *LabelTree* may swap C with A or B. If A is swapped with C then the pair $(n_{010}/n_{110}, n_{001}/n_{101})$ is compared and if B is

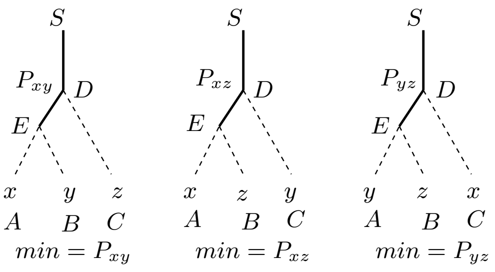


Figure 5. *LabelTree* **Procedure : Minimum pair-wise common passage probabilities are shown in bold**

swapped with C then the pair $(n_{100}/n_{110}, n_{001}/n_{011})$ is compared. These unrelated ratios continue to remain different after A and B misbehave.

To perform the check of lemma 3, *HTest* is used. *HTest* is a standard statistical hypothesis test which is used to test the difference between two proportions. Given the feedbacks of three receivers A, B, and C, the following contingency table is constructed.

	$Lost$	$Crossed$	$Total$
$Sample\ A$	n_{100}	n_{101}	n_{10*}
$Sample\ B$	n_{010}	n_{011}	n_{01*}
$Total$	$n_{100} + n_{010}$	$n_{101} + n_{011}$	$n_{10*} + n_{01*}$

A two-tailed test is performed with *null hypothesis* H_0 and *alternative hypothesis* H_1 defined as,

$$H_0 : \frac{n_{100}}{n_{101}} = \frac{n_{010}}{n_{011}} \qquad H_1 : \frac{n_{100}}{n_{101}} \neq \frac{n_{010}}{n_{011}}$$

HTest tests whether the difference between the two ratios n_{100}/n_{101} and n_{010}/n_{011} is statistically acceptable with respect to the given sample sizes of sample $A(n_{10*})$ and sample $B(n_{01*})$. In this work, we use *Fisher's Exact Test* as the representative statistical test. Fisher's exact test is a permutation test and outputs a *p-value* between 0 and 1. If the p-value is less than 0.05, null hypothesis H_0 can be rejected with 95% confidence, i.e., with 95% confidence one can say that feedbacks of either or both A and B are incorrect.

ICheck exploits the diversity of the multicast tree and the feedback data to overcome the weaknesses of lemma 3. Firstly, in each three receiver test, lemma 3 cannot check $0 \rightsquigarrow 1$ errors in C's feedbacks. However, since the algorithm tests several different three receivers sets, when a receiver x appears as C in one set, its feedbacks are not checked; but when it appears as A or B in another set, its feedbacks get checked (Figure 6). Secondly, the test of lemma 3 is weaker when passage probability of path SD, i.e. P_{abc} is very high. When a receiver x appears in one set

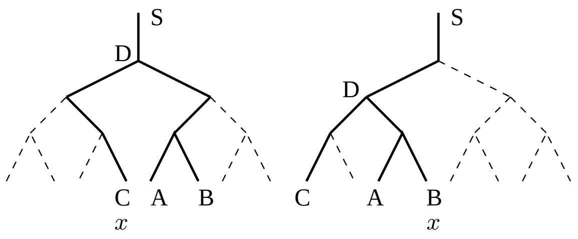


Figure 6. Functioning of *ICheck*

of three receivers, the least common parent (node D) may be close to the source S resulting in $P_{abc} \approx 1$. But when it appears in another set of three receivers, node D may be far from the source, resulting in $P_{abc} < 1$ (Figure 6).

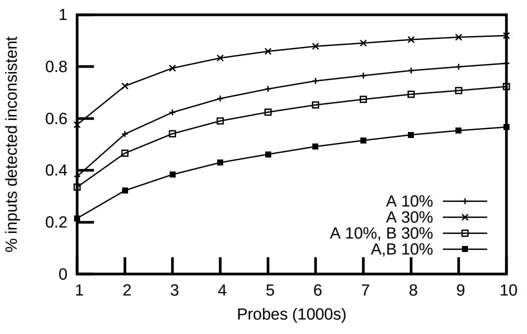
The complexity of *ICheck* varies with the number of three receiver sets checked. If k three receiver sets are tested, the complexity of *ICheck* is $O(Nk)$. As k increases, the integrity check becomes stronger. For the strongest check, when all $\binom{R}{3}$ three receiver sets are tested, k is $O(R^3)$.

5 Experiments

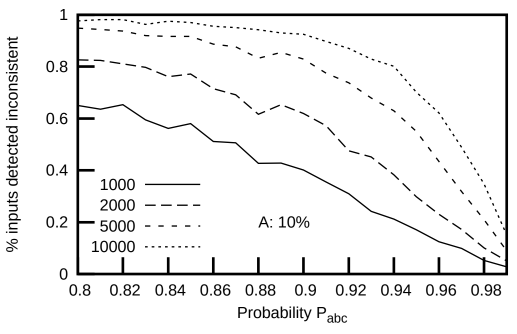
ICheck is a C program which we have implemented using the ideas discussed in the previous section. For Fisher's exact test, we use Algorithm 643 [7, 10] written in Fortran. To test the performance of *ICheck*, we have conducted experiments using model-based traces, NS traces and MBone traces. For Model-based traces, losses on each link are created using a time-invariant Bernoulli loss process. For NS simulations, losses on links occur due to buffer overflows at network nodes as the multicast probe competes with background TCP and exponential on-off UDP traffic. For MBONE traces, we use the traces of a multicast audio session which was run on the MBONE.

5.1 Model Simulation

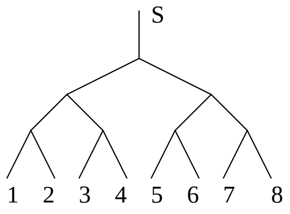
Model based simulations are used to study the performance of *ICheck* on a wide variety of inputs. The effectiveness of *ICheck* rests on the statistical test performed on three receiver sets. The factors from input data which influence these tests are *(i)* Number of probes *(ii)* Actual link loss rates in the multicast tree. These two factors work together to determine the sizes of two samples n_{10*} and n_{01*} which are compared. If these two samples are of small sizes and their size difference is significant, only large alterations of feedbacks can be detected by the statistical test. As the sizes of these samples grow, their size difference matters less and even small alterations of feedbacks are detected. Whenever the samples are of comparable sizes, the detection is stronger. As the number of probes grow, sample



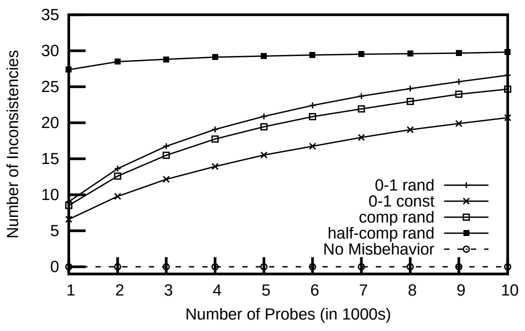
(a) Performance of the statistical test on different types of 3-receiver inputs



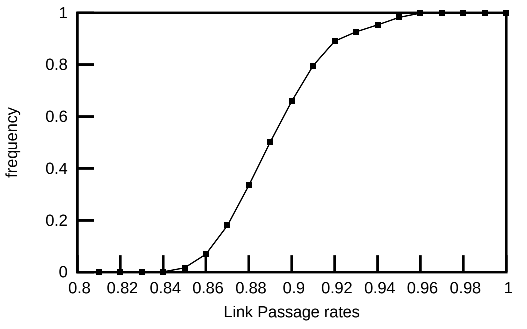
(b) Influence of P_{abc} in different types of 3-receiver inputs



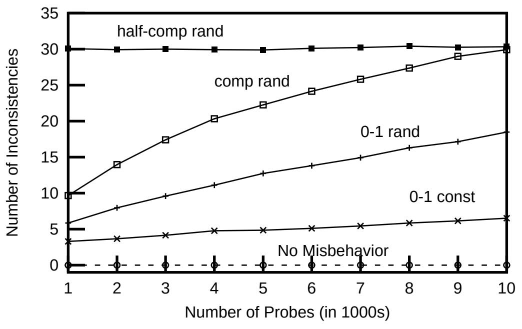
(c) 8-receiver tree



(d) Performance of *ICheck* on Model Based Traces



(e) CDF of link passage rates (NS simulation)



(f) Performance of *ICheck* on NS loss traces

Figure 7. Performance of *ICheck* Algorithm

sizes become large and eventually all alterations of feedbacks are detected.

Figure 7(a) shows the effectiveness of the statistical test performed by *ICheck* on different types of three receiver inputs. In this experiment, the three receiver topology of figure 3(a) was considered and the passage probabilities of all links were varied from 0.80 to 0.99 to yield an almost complete range of possible three receiver inputs which could be tested by *ICheck*. For each input the number of probes were varied from 1000 to $10,000$. The *y-axis* in Fig 7(a) plots the percentage of inputs where the statistical test successfully detects the inconsistency with 95% confidence for different misbehavior mechanisms :(i) 10% of receiver A's feedbacks are altered from 0 to 1 (ii) 30% of receiver A's feedbacks are altered from 0 to 1 (iii) 10% of receiver A's feedbacks and 30% of receiver B's feedbacks are altered from 0 to 1 (iv) 10% of both A and B's feedbacks are altered from 0 to 1. As the number of probes increase, the detection is stronger. When receivers misbehave with the same probability, the detection is slightly weaker since the ratios n_{100}/n_{101} and n_{010}/n_{011} are less far-apart as compared to when receivers misbehave with different probabilities. Fig 7(b) plots the same results as a function of the passage probability P_{abc} for the case where 10% of receiver A's feedbacks are altered from 0 to 1. P_{abc} is crucial compared to other link probabilities since it determines the amount by which the two ratios can get distorted, when feedbacks are altered. As P_{abc} grows larger, the two ratios change less and more probes are needed for detection.

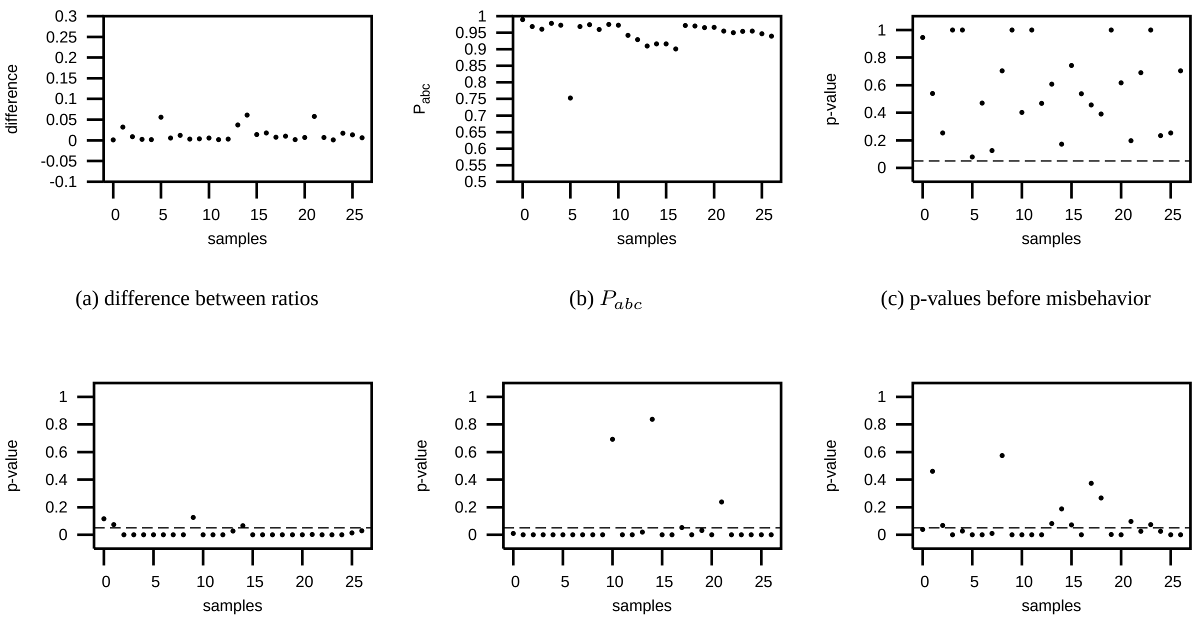
Next, the performance of *ICheck* was tested on general trees. Figure 7(d) shows the performance of *ICheck* on the 8-receiver complete binary tree shown in Fig 7(c). In this experiment, 1000 trees of the type in Fig 7(c) were generated with link passage probabilities varying uniformly from 0.80 to 0.99. For each tree, probes were simulated and loss traces obtained. In each trace, the following misbehavior mechanisms were introduced (i) Each receiver misbehaves from 0 to 1 with either 0%, 5%, 10% or 15% probability uniformly (0-1 rand). (ii) All receivers misbehave from 0 to 1 with with 10% probability (0-1 const) (iii) Each receiver complements 0%, 5%, 10%, or 15% of its feedbacks (comp rand) (iv) A random set of half of the receivers complement all their feedbacks (half-comp rand). In each trace, *ICheck* tested all the $\binom{8}{3} = 56$ three receiver sets. Figure 7(d) plots the average number of inconsistencies detected by *ICheck* for each misbehavior mechanism. When receivers complement their probes, the feedback matrix becomes quite inconsistent and several inconsistencies get detected. As the number of probes increase, the detection becomes stronger. The detection is weakest when all receivers misbehave with the same probability.

5.2 NS Simulation

For NS loss traces, the simulation parameters were setup as in the original work of MINC [2]. The 8-receiver complete binary tree shown in Fig 7(c) was considered. The bandwidth and propagation delay of each link were set 1.5Mbps and 10ms respectively. Each link was modeled as a FIFO queue with four-packet capacity. Node 0 sent 200 byte multicast probe packets with interpacket times chosen uniformly at random from 2.5 to 7.5 ms. We conducted 100 simulations and during each simulation, a variable amount of background traffic was introduced on each link in the tree using a random number of TCP and exponential on-off UDP flows. The probe losses observed by each receiver were used to generate the loss traces. Link passage rates in these simulations varied from 85% to 95%. Figure 7(e) shows the combined cummulative distribution function (CDF) of link passage rates for all links in the 100 simulations. For each trace four types of misbehavior mechanisms were considered as before. For each trace, *ICheck* tested all the $\binom{8}{3}$ three-receiver sets. Figure 7(f) plots the average number of inconsistencies detected by *ICheck* for each type of misbehavior mechanism. As observed before, when the number of probes increase, the detection becomes stronger.

5.3 MBONE traces

For MBone traces, we analyzed the WRN traces collected by [16] and publicly available at the web site [14]. These traces correspond to multicast audio sessions of World Radio Network(WRN). Each dataset is about an hour of trace in which receivers in the multicast group recorded the sequence number of audio packets they received at 80ms intervals. The following traces were analyzed: WRNSep19, WRNNov1, WRNNov13, WRNNov14, WRNNov28, WRNDec1 and WRNDec11 (topologies for all these traces are shown at [14]). From each dataset, 3 receivers which experienced sufficient losses were chosen. Their traces were made binary based on whether an audio packet was received or not and divided into batches of size $10,000$ each. This resulted in a total of 27 3-receiver loss traces of size $10,000$ each. Figure 8(a),(b) and (c) show the properties of these samples. Figure 8(a) shows the difference between the ratios n_{100}/n_{101} and n_{010}/n_{011}, figure 8(b) shows the common passage probability P_{abc} for each sample, and figure 8(c) shows the p-value obtained when the two ratios were given to $HTest$. Since there is no misbehavior, the p-values for samples lie above 0.05. Figures 8(d),(e) and (f) show the p-values calculated after three types of misbehavior mechanisms : 8(d) 10% of receiver A's feedbacks were altered from 0 to 1 8(e) 10% of receiver A's feedbacks and 30% of receiver



(a) difference between ratios (b) P_{abc} (c) p-values before misbehavior

(d) p-values: A 10% (e) p-values: A 10%, B 30% (f) p-values: A, B 15%

Figure 8. MBone Experiments

B's feedbacks were altered from 0 to 1 and 8(f) 15% of both receiver A and B's feedbacks were altered from 0 to 1. The p-values for most samples now lie below 0.05 indicating that one can conclude with 95% confidence that there is something wrong with the receiver feedbacks.

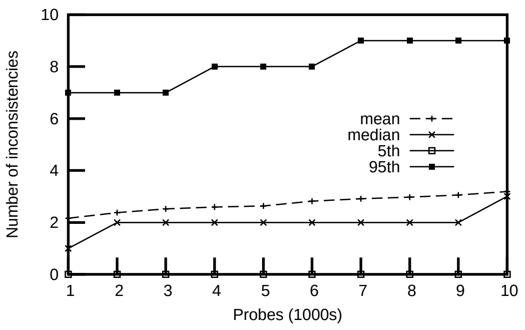
6 Discussion

6.1 Collusion

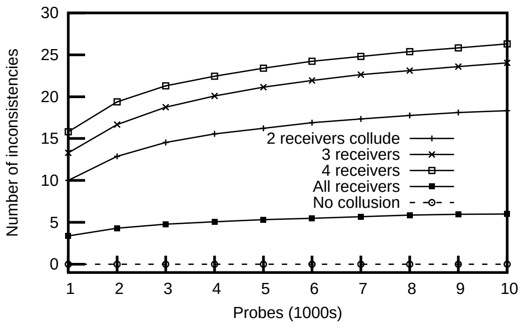
ICheck is a consistency check. It checks if the feedbacks of all receivers are consistent with respect to one another. Due to this reason it exhibits some resistance to collusion. Malicious receivers in a multicast tree can collude in small groups by reporting a 1 only when at least one member in the group received the probe packet. If two receivers have colluded together, their feedbacks may be consistent with respect to each other but inconsistent with respect to feedbacks reported by other receivers. Since *ICheck* tests different three receiver groups, when two receivers which have colluded together appear in different three receiver sets, their misbehavior could be detected. To illustrate this, we considered the following experiment. The 8-receiver tree of Fig 7(c) was considered and the link passage probabilities of all links were varied from 0.80 to 0.99 to generate 1000 random trees as before. In each tree, receivers 1 and 2 colluded with probability 50%, i.e., for 50% of the probes where at least one of the receivers received the probe, both of them reported 1. Fig 9(a) plots the number of inconsistencies detected by *ICheck* after collusion. It plots the mean, median, and 5th and 95th percentiles of the median. In spite of collusion *ICheck* detects inconsistencies because when receivers 1 and 2 appear in different three-receiver sets, their feedbacks get checked with the feedbacks reported by other receivers. For instance, if 1 and 2 appear in the same three-receiver set as in $(1,\ 2,\ 5)$, their misbehavior cannot be detected since the ratios tested by Lemma 3 would remain equal. But when the receiver set $(1,\ 3,\ 5)$ or $(2,\ 3,\ 6)$ is checked, inconsistency could be detected. Fig 9(b) plots the average number of inconsistencies detected by *ICheck* when a random set of receivers collude in each tree, i.e., for 50% of the probes where at least one receiver in the set received the probe, all of them reported 1. Even when all receivers in the tree collude, *ICheck* detects inconsistencies. In order to collude in a manner such that the entire feedback matrix rests consistent after collusion, receivers need to know the multicast tree topology and perform synchronized collusion in large groups.

6.2 Spatial Dependence

In multicast, losses occurring on different links can be dependent to each other. Spatial dependence of losses can particularly occur between sibling links (i.e., links which



(a) Receivers 1 and 2 collude (in tree Fig 7(c))



(b) Random group of receivers collude (in tree Fig 7(c))

Figure 9. Collusion

have a common parent) when multicast is implemented on overlay networks such as the MBONE. In such cases, sibling links may cross the same underlying physical link resulting in dependent losses. The test of Lemma 3 remains valid even in the presence of spatial dependence between sibling links, i.e., the ratios tested by Lemma 3 remain equal if there is no misbehavior. However, these ratios may become unequal if losses on non-sibling links are dependent to each other.

6.3 Comparison to Nonce based scheme

To avoid receiver misbehavior, a *nonce* bit can be sent in every probe packet which receivers need to return in case they report that they received the probe packet. The nonce based scheme has some disadvantages. Firstly, it requires the change of existing protocols used for measurement. Secondly, it results in receivers reporting more bits. This can pose a constraint when MINC is used with RTCP and receiver feedbacks must occupy only 5% of data bandwidth [3]. Thirdly, since the probe packet is a multicast packet, all receivers receive the same nonce bit. Thus collusion becomes very easy and a receiver can collude with any arbitrary receiver to report more 1's.

On the other hand, *ICheck* checks the integrity of receiver feedback data. Whenever receivers misbehave in a manner which destroys the consistency of feedback data, *ICheck* succeeds. However, due to the nature of statistical tests conducted, *ICheck* is necessary but not sufficient, i.e, if *ICheck* detects inconsistency, it means that there is something wrong with receiver feedbacks; but if it does not detect inconsistencies, it means that most likely receiver feedbacks are correct. Depending on the loss rates in the multicast tree and the amount of misbehavior, it is feasible, although less likely, that the resultant feedback matrix after misbehavior remains entirely consistent.

7 Conclusions

In order to use end-to-end network inference in a trustworthy manner, it is essential to verify the integrity of receiver feedbacks. In this paper, we showed how the MINC loss inference is affected by incorrect receiver feedbacks. We showed how the loss rates inferred by MINC change when incorrect feedbacks are received. Subsequently, we presented the *ICheck* algorithm which searches for loss rate inconsistencies that arise in erroneous feedback data. We presented the performance of *ICheck* on Model Based traces, NS traces and MBONE traces. Our experiments showed that *ICheck* successfully detects inconsistencies in the presence of different types of misbehavior mechanisms and even in the presence of collusion. *ICheck* does not require any knowledge of the multicast tree topology and thus does the affect the end-to-end nature of MINC loss tomography. The *ICheck* algorithm can be used in the phase before the MINC loss Inference to determine whether the inference based on the given feedbacks would most likely be trustworthy or not. In future, we shall investigate the problem of identification of receivers which have reported incorrect feedbacks.

References

[1] A. Adams, T. Bu, T. Caceres, N.G.Duffield, T. Friedman, J.Horowitz, F.Lo Presti, S.B. Moon, V. Paxson, and D. Towsley. The use of end-to-end multicast measurements for characterising internal network behavior. *IEEE Communications Magazine*, May 2000.

[2] R. Caceres, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions on Information Theory*, 7:2462–2480, Nov 1999.

[3] R. Caceres, N.G. Duffield, and T. Friedman. Impromptu measurement infrastructures using RTP. In *IEEE INFOCOM*, June 2002.

[4] Dawson Engler and Madanlal Musuvathi. Model-checking large network protocol implementations. *Network System Design and Implementation (NSDI)*, 2004.

[5] T. Friedman (ed.), R. Caceres (ed.), A. Clark (ed.), K. Almeroth, R. G. Cole, N. Duffield, K. Hedayat, K. Sarac, and M. Westerlund. RTP control protocol extended reports (RTCP XR). RFC 3611, Internet Engineering Task Force, November 2003.

[6] Sergey Gorinsky, Sugat Jain, and Harrick Vin. Multicast congestion control with distrusted receivers. *NGC*, 2002.

[7] Cyrus R. Mehta and Nitin R. Patel. ALGORITHM 643 FEXACT: A FORTRAN subroutine for Fisher's exact test on unordered r x c contingency tables. *ACM Trans. Math. Softw.*, 12(2):154–161, 1986.

[8] V. Paxson, A. K. Adams, and Matt Mathis. Experiences with NIMI. *Passive and Active Measurement workshop*, April 2000.

[9] Francesco Lo Presti, N. G. Duffield, J. Horowitz, and Don Towsley. Multicast-based inference of network-internal delay distributions. *IEEE/ACM Trans. Netw.*, 10(6):761–775, 2002.

[10] Mehta C. R. and Patel N.R. A network algorithm for performing fisher's exact test in rxc contingency tables. *Journal of the American Statistical Association*, 78:427–434, 1983.

[11] Sylvia Ratnasamy and Steven McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. *IEEE INFOCOM*, 1999.

[12] H. Schulzrine, S. Casner, R. Frederick, and V. Jacobson. RFC 1889: RTP : A Transport Protocol for Real-Time Applications, January 1996.

[13] B. Tierney and D. Gunter. Netlogger: A toolkit for distributed system performance tuning and debugging. *LBNL Tech Report*, 2002.

[14] MBone Traces. ftp://gaia.cs.umass.edu/pub/yajnik/.

[15] Andrew Whitaker, Richard S. Cox, and Steven D. Gribble. Configuration debugging as search: Finding the needle in the haystack. *6th Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[16] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the mbone multicast network. *Global Internet Conference*, Nov 1996.