# Timing-based Localization of In-Band Wormhole Tunnels in MANETs

Jinsub Kim*, Dan Sterne†, Rommie Hardy‡, Roshan K. Thomas†, and Lang Tong*
*ECE Dept., Cornell University. †Cobham Analytic Solutions. ‡U. S. Army Research Laboratory.
*{jk752, lt35}@cornell.edu, †{Dan.Sterne, roshan.thomas}@cobham.com, ‡rhardy@arl.army.mil

## ABSTRACT

The problem of localizing in-band wormhole tunnels in MANETs is considered. In an in-band wormhole attack, colluding attackers use a covert tunnel to create the illusion that two remote network regions are directly connected. This apparent shortcut in the topology attracts traffic which the attackers can then control.

To identify the nodes participating in the attack, it is necessary to determine the path through which victims' traffic is covertly tunneled. This paper begins with binary hypothesis testing, which tests whether a suspected path is carrying tunneled traffic. The detection algorithm is presented and evaluated using synthetic voice over IP (VoIP) traffic generated in a network testbed. After that, we consider multiple hypothesis testing to find the most likely tunnel path among a large number of candidates. We present a tunnel path estimation algorithm and its numerical evaluation using Poisson traffic. A main feature of the proposed algorithms is their robustness against the presence of chaff packets (possibly introduced to avoid detection), packet loss caused by unreliable wireless links, and clock skew at different nodes.

## Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection (e.g., firewalls)*

## General Terms

Security

Work in this paper was prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2- 0011, and was sponsored in part by National Science Foundation under Contract CCF-0635070 and Army Research Office MURI Program under award W911NF-08-1-0238. The first author was partially supported by Samsung Scholarship. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.


*WiSec'10,* March 22–24, 2010, Hoboken, New Jersey, USA.


## 1. INTRODUCTION

Mobile ad hoc networks (MANETs) rely on cooperative routing protocols in which ordinary nodes work together to form appropriate routes and forward traffic along them. The dynamic nature of the network topology mandates that routes be discovered and maintained continuously. A fundamental security issue is that a small number of compromised nodes may be able to manipulate these protocols to disrupt traffic throughout the network.

An example is the in-band *wormhole attack* [19], in which colluding nodes create the illusion that two remote regions are directly connected via a single-hop shortcut referred to as the *wormhole link*. The apparent shortcut undermines routing calculations and allows the attackers to attract and control traffic that would not flow through them otherwise. If optimally positioned, the attackers may be able to attract and control a large fraction of the network's traffic.

The wormhole attack requires two attacking nodes to serve as a pair of endpoints of the wormhole tunnel, and they covertly tunnel traffic between the regions by exploiting other unsuspecting nodes as traffic forwarders. The attack typically requires one or more colluding attackers that serve as application-layer waypoints along the tunnel path [19]. These waypoints stabilize the tunnel by breaking the tunnel path into segments each having a route that is short enough to be unaffected by the presence of the wormhole link. See Section 2.1 for the discussion of a specific example.

In this paper, we consider the problem of localizing (*i.e.*, identifying) the covert tunnel path based on packet timing information as a means of identifying the attacking nodes, especially those that serve as tunnel waypoints. We assume that one or more of the victim nodes suspect the presence of a wormhole, and thus initiate a sequence of tests to localize the tunnel. This process, described below, is based on the premise that if the forwarded traffic has delay constraints (*e.g.*, VoIP and other time sensitive applications), then transmission times at nodes along the tunnel path exhibit strong temporal correlations, which allow the detection of the presence of tunneled traffic.

The use of timing information does not need to be exclusive in practice; there may be other sources of evidence (*e.g.*, the knowledge of packet headers) that can be incorporated to enhance the localization performance. In this paper, however, we will focus entirely on a timing based approach, motivated by the need to understand the value of timing in detection and the fact that packet headers and other auxiliary information may be unavailable due to the encryption of forwarded traffic.

## 1.1 Summary of Contribution and Limitations

This paper presents timing-based algorithms for localizing in-band wormhole tunnels in MANETs. To our best knowledge, the proposed approach is the first directed at identifying covert tunnels in their entirety, including the colluding relay nodes that are required to prevent wormhole tunnels from collapsing [19]. Furthermore, it is applicable to both the self-contained and extended in-band wormholes [19].

As the simplest case, we first present a detection algorithm aimed at determining whether a suspected path is the true tunnel path. This detector has its origin in [15] but includes a nontrivial extension to deal with clock skew present in MANETs. Then, we present a path estimation algorithm aimed to find the most likely tunnel path among a large number of candidates. The proposed algorithms are intended to be used in conjunction with other existing techniques that detect the likely presence of a wormhole attack and identify the endpoints of the suspected wormhole link. Our algorithms are intended to validate such suspicion and identify the correct tunnel path if an attack is present. We describe a simple conceptual model of how these components can be integrated into a tunnel localization system. The proposed algorithms are evaluated using synthetic VoIP traffic generated in a network testbed and Poisson traffic, and the results are quite promising. Both algorithms have linear complexity with respect to the number of samples used.

The proposed algorithms are robust against various practical networking uncertainties, especially the presence of timing jitter and chaff packet transmissions. The algorithms are non-parametric in the sense that they do not require the knowledge of probability distributions of the timing observations although some of the analytical results (Theorem 1) and the numerical results are obtained under specific probabilistic models. Indeed, the synthetic VoIP traffic, used for evaluation, is generated from a practical emulation system that implements a suite of realistic networking protocols.

The main limitation of the proposed algorithms is the requirement of persistent measurements and the timing constraints. In particular, our algorithms apply to those scenarios in which a relatively long sequence (from 100s to 1000s) of packets is passed through a wormhole tunnel, and each packet is subject to a delay constraint at forwarding nodes. Such limitations make the technique appropriate for time sensitive applications such as VoIP, but may not be applicable for the detection of tunneling of individual packets.

The use of timing alone also limits the localization performance, which was discussed in [15]. Specifically, the flow tunneled through the wormhole has to be sufficiently strong. In other words, a timing-based localization scheme can be defeated if the attacker artificially inserts a large enough number of dummy (chaff) transmissions. However, this may not be a severe limitation because the attacker may not have control of all nodes in the tunnel, and the transmission of a large number of dummy packets may reveal the presence of an attack.

## 1.2 Related Work

Most existing techniques for detecting wormhole attacks in MANETs concern out-of-band wormholes, in which attackers connect the purported neighbors via an extra RF channel or wireline network not accessible to other nodes. These attacks do not utilize covert tunnels. The concept of an out-of-band wormhole in ad hoc networks was introduced by Hu [17], who outlines temporal and geographic countermeasures designed to detect the remote forwarding of packets. Hu describes packet leashes, which attempt to restrict the maximum transmission distance of a packet. In this scheme, packets that arrive through wormhole paths will be received outside a tightly synchronized time window and can be treated by the recipient as invalid. Other distance bounding approaches for out-of-band wormholes are described by Lazos [20], Khalil [18], and Adjih [5]. Buttyán [9] proposes techniques for detecting out-of-band wormholes based on statistical changes to neighbor hop counts and path lengths. Gorlatova [13] describes the detection of out-of-band wormholes in an OLSR network [10] by analyzing the power spectral density of periodic HELLO messages received from neighboring nodes. If the HELLOs have arrived via a wormhole, the associated delay, even if quite small, is said to smear the HELLO message time series. Awerbuch [6, 7] proposes the On-Demand Secure Byzantine Routing protocol (ODSBR), and describes its ability to defend against various attacks, including out-of-band wormholes. ODSBR mechanisms do not detect wormholes per se; instead they detect packet dropping that has been applied to traffic traveling through wormholes.

Research concerning in-band wormholes has focused on identifying attacking nodes at tunnel endpoints. In-band wormhole attacks were first described in detail by Kruus [19]. Kruus proposes detecting these attacks and identifying attackers at wormhole tunnel endpoints by collecting roundtrip packet loss and delay measurements for short paths throughout the network and regionally correlating those measurements that are unexpectedly high. Sterne [22] extends this approach by using opportunistic voting to counter the threat that Byzantine nodes may deliberately introduce path measurement errors that act as false accusations against honest nodes. Zheng [25] also examines the detection of in-band wormholes by collection of round trip delay measurements but applies more elaborate statistical analysis techniques to these measurements to distinguish wormhole-induced delays from network congestion. Unlike our techniques, none of these identifies colluding relay nodes. Furthermore, these techniques are primarily applicable to the self-contained form of in-band wormhole (see Section 2.1).

The mathematical techniques adopted in this paper belong to the family of traffic analysis [12] aimed at drawing inference from timing patterns. The genesis of our approach may be traced to the seminal work by Donoho *et. al.* [11] where the authors provided insights into the use of timing information to detect stepping stone attacks. It is Blum, Song, and Venkataraman [8] who provided a mathematically rigorous approach to the detection of a sequence of packets subject to delay constraints. Their approach is later generalized by He and Tong [14–16] to deal with the presence of chaff in the timing measurements. The mathematical theory behind the detection of information flow was presented in [15] where the fundamental limits of flow detection using timing measurements and the forms of detectors are presented. Motivated by [15], this paper provides specific implementations for the wormhole tunnel localization in practical MANETs, including a new technique to deal with synchronization and the tunnel path estimation algorithm. Another relevant technique is the use of the concept of water marking by Wang and Reeves. See [23] and references therein. Such techniques are vulnerable when the

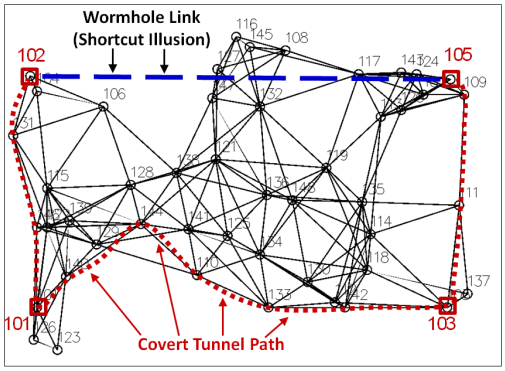



Figure 1: Self-contained In-Band Wormhole

adversary can significantly perturb the timing information, as it is possible in this case.

### 1.3 Organization

This paper is organized as follows. In Section 2, we introduce the attack model, the main assumptions adopted in this paper, the wormhole tunnel localization system, and the mathematical model of a wormhole attack. In Section 3, we introduce the algorithm aimed at determining whether a suspected path is the true tunnel path, and present analytical and experimental results. Section 4 proposes the tunnel path estimation algorithm, which finds the most likely tunnel path among a large number of candidates, and presents its numerical evaluation. Section 5 contains discussion about the results and possible future work. Section 6 concludes the paper with remarks on its contributions.

## 2. WORMHOLE ATTACK AND LOCALIZATION SYSTEM

### 2.1 An Example of a Wormhole Attack

An example of a self-contained in-band wormhole in a 48-node MANET that uses the OLSR routing protocol is shown in Fig. 1. This attack involves four attacking nodes positioned in a roughly rectangular arrangement around the periphery of the network. These nodes, 101, 102, 103, and 105, are highlighted in the figure by four small surrounding squares. The wormhole link created by the attack (the illusory shortcut) is shown as a dashed blue straight line between attacking nodes 102 and 105 near the top of the figure. The wormhole tunnel path is shown in the figure as a dotted red line connecting four attacking nodes. To make it appear that nodes 102 and 105 are directly connected, 102 covertly sends into the tunnel copies of all of its outgoing one-hop packets, including OLSR HELLO (neighbor sensing) messages, other broadcast packets, and forwarded packets sent to 102's layer 2 address. This allows such packets to reach node 105 despite the fact that 105 is more than one hop from 102. Node 105 similarly copies into the tunnel outbound one-hop packets that would reach 102 if these two nodes were directly connected. This creates the illusion that nodes 102 and 105 are directly connected and causes many nodes on the left and right sides of the figure to believe that the shortest path to the opposite side of the network is via nodes 102 and 105, and to route their traffic to those attacking nodes for forwarding.

Attacker nodes 101 and 103, at the bottom left and bottom right, serve as the application-layer waypoints needed to stabilize routing through the tunnel, as mentioned above. When node 102 sends a packet into the tunnel, it encapsulates the packet and sends it through a tunnel segment that terminates at node 101, the closest waypoint. Packets sent into this tunnel segment are addressed at the network layer to node 101. After a packet emerges from the segment tunnel at node 101 and is de-encapsulated, node 101 re-encapsulates it and copies into another segment tunnel that terminates at the next way point, node 103. Similarly, node 103 pushes the packet through the final tunnel segment to node 105. Note that nodes along the tunnel path, other than the colluding waypoints, have no knowledge of the fact that they are supporting this covert tunnel. For example, because of encapsulation, packets forwarded by node 133 (near the bottom of the figure) appear to be ordinary packets sent by node 101 to 103.

In the extended in-band wormhole attack [19], rather than copying their own one-hop packets into the tunnel, nodes 102 and 105 copy into the tunnel one-hop packets promiscuously overhead emanating from one or more of their real neighbors. When these packets emerge from the far end of the tunnel, the receiving attacker rebroadcasts them. This creates the illusion that the attackers' own neighbors are directly connected. For example, nodes 106 (a neighbor of 102) and 109 (a neighbor of 105) will hear each other's transmissions and believe they are directly connected. This form of wormhole can be used to create a mesh of wormhole links between their respective sets of neighbors.

### 2.2 Practical Assumptions

We envision our tunnel localization algorithms as being incorporated into a *cooperative intrusion detection system* [24]. In such a system, nodes throughout a MANET are recruited to serve as intrusion detection sensors. To support tunnel localization, we require that each recruited node keeps logs of recent packet transmission times and destination addresses and transfer excerpts from these logs on demand to a designated correlation node when a wormhole attack is suspected. We also assume that all packet transmission logs, including those submitted by attacking nodes, are correct. Although a cooperative intrusion detection system that is deployed operationally must account for the possibility that attacking nodes may deliberately report erroneous transmission logs, addressing that threat is beyond the scope of this paper.

To an attacker, the primary value of a wormhole attack is that it attracts traffic, which the attacker can control at an opportune time in the future, e.g., by discarding, delaying, or damaging packets before forwarding them. Consequently, a wormhole that persists is of greater threat than a wormhole that exists momentarily or intermittently, because it allows the adversary to lie in wait. As a result, for the defender, detecting the onset of a wormhole attack immediately is much less important than detecting continuing wormhole activity reliably and accurately. In this regard, wormholes and other attacks on routing protocols pose a different kind of threat than host-penetration attacks in which a single malicious packet may cause substantial damage and must be detected immediately. In this vein, we make the simplifying assumption that a wormhole that poses a significant threat will persist and that its covert tunnel path will remain stable for at least one period of sufficient duration to log the

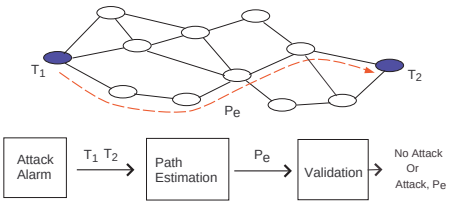



Figure 2: In-band Wormhole Tunnel Localization System: If attack is detected, the Attack Alarm block produces suspected tunnel endpoints $T_1$ and $T_2$. Then, the Path Estimation block gives the most likely tunnel path $P_e$, and the Validation block checks whether $P_e$ is being used as a tunnel path and makes a decision.

number of packet transmission events required by our tunnel localization algorithm.

## 2.3 Tunnel Localization System

Our conceptual model for an in-band wormhole tunnel localization system combines our localization algorithms with other techniques. As illustrated in Fig. 2, the localization system consists of three functional blocks: *Attack Alarm*, *Path Estimation*, and *Validation*.

For the Attack Alarm block, we assume that by using an existing technique, the presence of a wormhole attack can be detected by victims whose traffic travels through the wormhole link. For example, victims may be able to tell that an attack is underway because of the statistical distribution of round-trip times measured through paths that utilize the wormhole link [19, 25], the power spectral density of inter-Hello message arrival times received through the link [13], or other indicators. We further assume that such techniques will also identify the endpoints of the wormhole link. For a self-contained wormhole, which we will focus on here for simplicity, these nodes are also the tunnel endpoints. So when an attack is detected, the Attack Alarm block identifies tunnel endpoints and initiates the Path Estimation block.

The Path Estimation block employs the tunnel path estimation algorithm presented in Section 4. Initiated by the Attack Alarm block, this block estimates the most likely tunnel path among all possible paths between two suspected endpoints.

The Validation block receives the tunnel path estimate from the Path Estimation block, and uses the detection algorithm proposed in Section 3 to check whether the estimated path is being used as an in-band wormhole tunnel. If the estimated path is judged to be innocent, then the Validation block declares 'no attack'; otherwise, it declares 'attack' and identifies the tunnel path.

In the localization system, the path estimation algorithm is used earlier than the validation algorithm for a single path. However, we deal with the single path validation problem first, in Section 3, because it gives the intuition behind the tunnel path estimation algorithm.

## 2.4 Mathematical Model

### 2.4.1 Notation

The transmission timing at a set of nodes is modeled as point processes. We use uppercase bold letters (*e.g.*, $\mathbf{S}$) to denote point processes and the corresponding lowercase bold letters (*i.e.*, $\mathbf{s}$) to denote their realizations. For a point process $\mathbf{S}$, we use $S(k)$ to denote the random variable corresponding to the $k$th transmission epoch, and $s(k)$ its realization. Given two realizations of point processes $(a_1,\ a_2, \ldots)$ and $(b_1,\ b_2, \ldots)$, $\bigoplus$ is the *superposition operator* defined as $(a_k)_{k=1}^{\infty} \oplus (b_k)_{k=1}^{\infty} = (c_k)_{k=1}^{\infty}$, where $c_1 \le c_2 \le \cdots$ and $\{a_k\}_{k=1}^{\infty} \cup \{b_k\}_{k=1}^{\infty} = \{c_k\}_{k=1}^{\infty}$. Given a realization $\mathbf{s}$, we use $\mathbb{S}$ to denote the set of elements in this realization

### 2.4.2 Information flow and Observation Model

We assume that the MANET carries information flows, and the wormhole attracts certain flows through its tunnel. We assume that these flows have delay constraints such that packets of such flows must be forwarded by intermediate nodes within certain deadlines. The notion of information flow with a bounded delay constraint can be formally defined as below.

*Definition 1.* Let $\mathbf{F}_i$ denote the point process corresponding to the transmission epochs at relay node $R_i$. Then the sequence of processes $(\mathbf{F}_1, \ldots, \mathbf{F}_n)$ forms an *information flow* with bounded delay $\Delta$ if for every realization $\mathbf{f}_i$ $(i = 1, \ldots, n)$, there exist bijections $g_i : \mathcal{F}_i \to \mathcal{F}_{i+1}$ $(i = 1, \ldots, n-1)$ such that $0 \le g_i(s) - s \le \Delta$ for all $s \in \mathcal{F}_i$.

The bijection $g_i$ maps the transmission timing of a packet in $R_i$ to that of the same packet in $R_{i+1}$. The bijection condition means *packet conservation*, and $g_i(s) - s \in [0,\ \Delta]$ ensures *causality* and *a maximum delay* $\Delta$.

In practice, a node can multiplex different traffic in its transmissions. It can also introduce dummy transmissions to confuse the intrusion detection system. In addition, if a packet is dropped in the middle of the path, then the packet is not a part of an information flow. Therefore, timing traces at monitored nodes may include an information flow and some other transmissions to which we refer as *chaff noise*.

Under the hypothesis that a set of nodes $R_i$ forms a wormhole tunnel, the observed transmission epochs $\mathbf{S}_i$ at $R_i$ will then be a superposition of an information flow $\mathbf{F}_i$ and chaff noise $\mathbf{W}_i$:

$$\begin{array}{ll} \mathbf{S}_i = \mathbf{F}_i \oplus \mathbf{W}_i, & i = 1, \ldots, n, \\ \mathbf{F}_{i+1} = g_i(\mathbf{F}_i) & i = 1, \ldots, n-1. \end{array} \tag{1}$$

Note that chaff noise is not subject to any constraints on information flows and can be correlated with the flows.

In this paper, we mainly consider two problems, single path validation and tunnel path estimation. Their mathematical formulations are given in the beginning of Section 3 and Section 4.

# 3. SINGLE PATH VALIDATION

This section presents the detection algorithm for the Validation block, which detects the presence of a wormhole tunnel on a suspected path. The algorithm also provides the intuition behind the tunnel path estimation algorithm proposed in Section 4.

## 3.1 Single Path Validation Problem

Suppose that we are interested in detecting whether a sequence of nodes, $R_1, R_2, \ldots, R_n$, forms an in-band wormhole tunnel. Let $\mathbf{S}_i$ $(i = 1, \ldots, n)$ be the process of transmission timestamps of node $R_i$. By observing $\mathbf{S}_i$ $(i = 1, \ldots, n)$ for some time $t$ $(t > 0)$, test the following hypotheses:

$$\begin{array}{ll} \mathcal{H}_0: & \mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_n \text{ are jointly independent} \\ \mathcal{H}_1: & (\mathbf{S}_i)_{i=1}^{n} \text{ contains an information flow} \end{array} \tag{2}$$

We note that the above two hypotheses are not complimentary in general. In general, a flow may travel a subset of relay nodes, say only $R_1$, $R_2$, and $R_3$. In that case, only timing at those three nodes would satisfy (1). In practice, one will need to execute a sequence of the tests of the form (2), starting with validating first whether $R_1$ and $R_2$ carry a flow. If positive, we then verify whether $R_1$, $R_2$, and $R_3$ carry a flow and so on.

## 3.2 Fundamental Limit on Consistent Detection

Using timing information alone has its limit in detecting the presence of an information flow traveling through a set of relay nodes. Intuitively, even for any realization of jointly independent transmission epochs ($\mathcal{H}_0$ in (2)), the decomposition of the form (1) is possible if the rate of the information flow is sufficiently low. Thus the detectability of the wormhole from timing information hinges on the strength of the flow being sufficiently strong. We therefore need the notion of *chaff-to-traffic ratio* (CTR) under $\mathcal{H}_1$.

*Definition 2.* [15] Given the realizations of an information flow $(\mathbf{f}_i)_{i=1}^n$ and chaff noise $(\mathbf{w}_i)_{i=1}^n$, the *chaff-to-traffic ratio* (CTR) is defined as

$$\mathrm{CTR}(t) \triangleq \frac{\sum_{i=1}^{n} |\mathcal{W}_i \cap [0,\, t]|}{\sum_{i=1}^{n} |(\mathcal{F}_i \cup \mathcal{W}_i) \cap [0,\, t]|}, \tag{3}$$

$$\mathrm{CTR} \triangleq \limsup_{t\to\infty} \mathrm{CTR}(t)$$

where $|\mathcal{W}_i \cap [0,\, t]|$ is the number of time epochs corresponding to the chaff packets at node $R_i$ within the time period $[0, t]$ and $|(\mathcal{F}_i \cup \mathcal{W}_i) \cap [0,\, t]|$ the total number of transmission epochs at node $R_i$ during the same time.

It was shown in [15] that flows with CTR greater than a certain value can be hidden to avoid the detection. We therefore need the notion of Chernoff-consistency [21].

*Definition 3.* Let $\delta_t$ be a detector that uses all timing data up to time t. The detector $\delta_t$ is called *r-consistent* ($r \in [0,\ 1]$) if it is Chernoff-consistent for all the information flows with CTR bounded almost surely by $r$. In other words, the false alarm probability $P_F(\delta_t)$ and the miss probability $P_M(\delta_t)$ satisfy the following:

1. $\lim_{t\to\infty} P_F(\delta_t) = 0$ for any $(\mathbf{S}_i)_{i=1}^n$ under $\mathcal{H}_0$;
2. $\sup_{(\mathbf{S}_i)_{i=1}^n \in \mathcal{P}} \lim_{t\to\infty} P_M(\delta_t) = 0$, where

$$\mathcal{P} = \{(\mathbf{S}_i)_{i=1}^n :\ (\mathbf{S}_i)_{i=1}^n \text{ contains an information flow, and } \limsup_{t\to\infty} \mathrm{CTR}(t) \le r \text{ a.s.}\}.$$

The *consistency* of a detector is defined as the supremum of $r$ such that the detector is $r$-consistent.

Consistency of the detector is the supremum of the fraction of chaff packets the detector can tolerate. Therefore, higher consistency means that the detector is more robust to chaff noise. In what follows, we will present a detection algorithm and establish its Chernoff consistency.

## 3.3 Background: Minimum CTR Flow Detection

The structure of the proposed detector is based on a threshold test on a lower bound $\widehat{\mathrm{CTR}}(t)$ on the true $\mathrm{CTR}(t)$ as defined in (3). Specifically, the proposed detector takes the following form

$$\begin{cases} \text{declare } \mathcal{H}_0 \text{ (no attack)} & \text{if } \widehat{\mathrm{CTR}}(t) > \tau \\ \text{declare } \mathcal{H}_1 \text{ (attack)} & \text{if } \widehat{\mathrm{CTR}}(t) \le \tau \end{cases} \tag{4}$$

To establish the Chernoff consistency of the above test, we use the minimum CTR statistics. Specifically, given the observed transmission epochs $(\mathbf{s}_i)_{i=1}^n$, we construct the test statistic by the following optimization

$$\widehat{\mathrm{CTR}}(t) \triangleq \min_{\mathbf{f}_i, \mathbf{w}_i : \mathbf{s}_i = \mathbf{f}_i \oplus \mathbf{w}_i \sim \mathcal{H}_1} \frac{\sum_{i=1}^{n} |\mathcal{W}_i \cap [0,\, t]|}{\sum_{i=1}^{n} |(\mathcal{F}_i \cup \mathcal{W}_i) \cap [0,\, t]|} \tag{5}$$

where $\mathbf{s}_i = \mathbf{f}_i \oplus \mathbf{w}_i \sim \mathcal{H}_1$ stands for the constraint that $\mathbf{s}_i$ carry a flow $\mathbf{f}_i$ with bounded delay as defined in $\mathcal{H}_1$.

We will delay the discussion of the ways of obtaining the above optimization with a linear complexity algorithm to Section 3.4. For now, we assume that the above optimization can be easily obtained and present a theoretical justification for the detector given in (4).

In [15], assuming that the timing epochs are Poisson processes, it is shown that, under $\mathcal{H}_0$,

$$\exists\ \tau_o \in (0,1) \quad \text{s.t.} \quad \lim_{t\to\infty} \widehat{\mathrm{CTR}}(t) = \tau_o \quad \text{almost surely} \tag{6}$$

Furthermore, under $\mathcal{H}_1$, if CTR is less than $\tau_o$ almost surely, then

$$\limsup_{t\to\infty} \widehat{\mathrm{CTR}}(t) \le \mathrm{CTR} < \tau_o \quad \text{almost surely} \tag{7}$$

Therefore, if we choose the detection threshold in (4) as $\tau_o - \epsilon$ with sufficiently small positive $\epsilon$, then the detector is $\tau_o - \epsilon$ consistent. What is left is a way to obtain $\widehat{\mathrm{CTR}}(t)$ in (5).

## 3.4 Computation of Minimum CTR

The algorithm that computes the above statistic is first proposed in [15]. Referred to as Multi-Bounded Delay Relay (MBDR), this algorithm partitions optimally the received traces $\mathbf{s}_i$ into the flow components $\mathbf{f}_i$ and the chaff components $\mathbf{w}_i$, where the flow components satisfy the bounded delay constraint. Here we present MBDR assuming first that there is no timing error in the transmission epoch measurements. MBDR works as follows:

Given the measurements $(\mathbf{s}_i)_{i=1}^n$:

1. Match every packet at time $t_1$ in $\mathbf{s}_1$ with the first unmatched packet $t_2$ in $[t_1,\ t_1 + \Delta]$ in $\mathbf{s}_2$, conditioned on that $t_2$ has a match in $\mathbf{s}_3$.
2. For $i = 2, \ldots,\ n-1$, match the packet $t_i$ in $\mathbf{s}_i$ with the first unmatched packet $t_{i+1}$ in $[t_i,\ t_i + \Delta]$ in $\mathbf{s}_{i+1}$, conditioned on that $t_{i+1}$ has a match in $\mathbf{s}_{i+2}$ (assume every packet in $\mathbf{s}_n$ has a match).
3. After trying to match all the packets in $\mathbf{s}_1$, label all the unmatched packets as chaff.

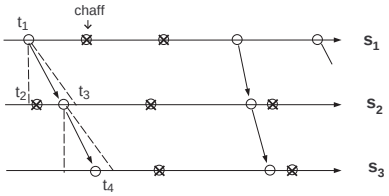



Figure 3: MBDR

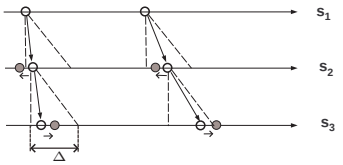



Figure 4: Damage from Clock Skews.

For example, consider the two-hop case illustrated in Fig. 3. To match $t_1 \in \mathcal{S}_1$, MBDR first tries to find a match for $t_2$. However, MBDR cannot match $t_1$ to $t_2$, because $t_2$ has no match in $\mathbf{s}_3$. Then, MBDR tries to find a match for $t_3 \in \mathcal{S}_2$, which is the next unmatched packet in $[t_1,\ t_1 + \Delta]$ in $\mathbf{s}_2$. Since $t_3$ can be matched with $t_4 \in \mathcal{S}_3$, $t_1$ is matched with $t_3$. If $t_3$ does not have a match in $\mathbf{s}_3$, MBDR will try to match $t_1$ with the next unmatched packet in $[t_1,\ t_1 + \Delta]$ in $\mathbf{s}_2$. If there are no more packets left in that interval, MBDR will label $t_1$ as chaff.

For implementation of MBDR, please refer to Table 5 in [15]. The complexity of MBDR is $O(n^2|S_1|)$, which is linear with repect to the number of observations [15].

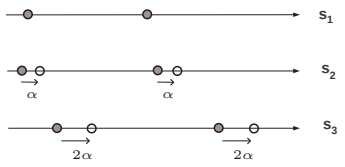
## 3.5 Minimum CTR Detection with Timing Synchronization

In this section, we introduce Minimum CTR Detection with Timing Synhronization (MCTRD-TS), an in-band wormhole tunnel detection algorithm robust to clock skew.

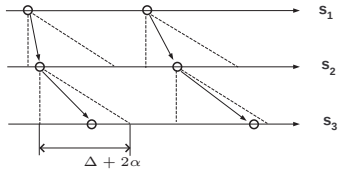
Clock skew can severely degrade the operation of MBDR. Fig. 4 illustrates an example of such damage. The empty circles represent the realizations of an information flow under the perfect clock synchronization assumption, and grey circles in $\mathbf{s}_2$ and $\mathbf{s}_3$ represent the measurements with the presence of clock skew. The arrows show how the clocks of node 2 and node 3 are different from node 1. Based on errorless measurments, MBDR should claim that there is no chaff. However, the measurements with timing errors make MBDR falsely declare that all packets are chaff. This example shows the need to take care of clock skew.

Because unrestricted clock skew would make the problem intractable, we suppose that clock differences between nodes are bounded by $\alpha$. Given the measurements with unknown timing errors, it is impossible to calculate the exact value of $\widehat{\mathrm{CTR}}(t)$. However, if the measurements are adjusted accordingly, we can still use them for detection.

Fig. 5 describes our approach with a two-hop example. Grey circles are the measurements with timing errors. First, we increase every timestamp in $\mathbf{s}_i$ by $(i-1)\alpha$ and denote the modified measurements by $(\overline{\mathbf{s}}_i)_{i=1}^3$. If $\underline{\mathbf{S}}_1$, $\underline{\mathbf{S}}_2$, and $\underline{\mathbf{S}}_3$ are independent point processes, then so are $\overline{\mathbf{S}}_1$, $\overline{\mathbf{S}}_2$, and $\overline{\mathbf{S}}_3$.




(a) Step 1




(b) Step 2

Figure 5: MCTRD-TS

Table 1: Minimum CTR Detection with Timing Synhronization (MCTRD-TS)

MCTRD-TS($\mathbf{s}_1, \ldots, \mathbf{s}_n, \Delta$):

for $i = 1 : n$
  for $j = 1 : |\mathcal{S}_i|$
    $s_i(j) \leftarrow s_i(j) + (i-1)\alpha$
  end
end
$\widehat{\mathrm{CTR}} \leftarrow$ MBDR($\mathbf{s}_1, \ldots, \mathbf{s}_n, \Delta + 2\alpha$)

return $\begin{cases} \mathcal{H}_1 & \text{if } \widehat{\mathrm{CTR}} \le \tau \\ \mathcal{H}_0 & \text{o.w.;} \end{cases}$

On the other hand, if $(\mathbf{s}_i)_{i=1}^3$ were drawn from $\mathcal{H}_1$, then the above adjustment recovers the causality of information flows, which could have been broken by clock skew. In addition, it can be easily checked that, after the adjustment, information flows satisfy the delay constraint $\Delta + 2\alpha$. Therefore, we can regard $(\overline{\mathbf{s}}_i)_{i=1}^3$ as our new measurements without timing errors, in which transmission delay is bounded by $\Delta + 2\alpha$. Based on this argument, MCTRD-TS with threshold $\tau$ works as follows: Given the measurements $(\mathbf{s}_i)_{i=1}^n$:

1. For $i = 2, \ldots, n$, increase every timestamp in $\mathbf{s}_i$ by $(i-1)\alpha$. Denote the modified measurements by $(\overline{\mathbf{s}}_i)_{i=1}^n$.
2. Apply MBDR with delay constraint $\Delta + 2\alpha$ to the modified measurements $(\overline{\mathbf{s}}_i)_{i=1}^n$, and calculate the test statistic $\widehat{\mathrm{CTR}}(t)$.
3. If $\widehat{\mathrm{CTR}}(t) > \tau$, declare $\mathcal{H}_0$ (no attack); otherwise, declare $\mathcal{H}_1$ (attack).

Implementation of MCTRD-TS is given in Table 1. Its computational complexity is same as that of MBDR, $O(n^2|\mathcal{S}_1|)$, which is linear with respect to the number of observations. The following states the consistency of MCTRD-TS.

*Theorem 1.* Assume that $(\mathbf{S}_i)_{i=1}^n$ under $\mathcal{H}_0$ are Poisson processes. Let $\tau_o$ be the value to which $\widehat{\mathrm{CTR}}(t)$ converges almost surely under $\mathcal{H}_0$. Then, for $\tau$ less than $\tau_o$, MCTRD-TS with threshold $\tau$ is $\tau$-consistent.

*Sketch of Proof:* Denote MCTRD-TS with threshold $\tau$ by $\delta_t$. Under $\mathcal{H}_0$, $\tau < \tau_o$ and (6) imply that $\lim_{t\to\infty} P_F(\delta_t) = 0$ for $(\mathbf{S}_i)_{i=1}^n$ under $\mathcal{H}_0$.

Table 2: Simulation Parameters

| | |
|---|---|
| $n$ | the number of processes |
| $\lambda$ | the rate of $\mathbf{S}_i$ $(i = 1, \ldots, n)$ |
| $\alpha$ | maximum clock difference |
| $\Delta$ | maximum delay |
| $f_c$ | CTR of the traffic under $\mathcal{H}_1$ |

Under $\mathcal{H}_1$, if CTR $\leq \tau$ almost surely, then (7) implies that $\lim_{t\to\infty} P_M(\delta_t) = 0$. Therefore, $\delta_t$ is $\tau$-consistent. ■

Furthermore, from theorem 3.2 in [15], it can be shown that $\tau_o$ is the supremum of consistency we can achieve by adjusting the threshold of MCTRD-TS.

Theorem 1 characterizes the detection performance and the limit of MCTRD-TS. Under the Poisson assumption, we can set $\tau$ to be $\tau_o - \epsilon$ for small positive $\epsilon$ and achieve $(\tau_o - \epsilon)$-consistent detector. Even in practical situations, $\tau_o$ in theorem 1 can be a good standard for a threshold. Experimental results in Section 3.7 address that $\tau_o$ is a lower bound for $\widehat{\text{CTR}}(t)$ of VoIP traffic under $\mathcal{H}_0$, when $t$ is sufficiently large. Hence, even for VoIP data, setting $\tau$ to be $\tau_o$ gives us a $\tau_o$-consistent detector.

Suppose that the maximum allowable false alarm probability $\kappa$ is given and we aim to minimize the miss detection probability. If we can acquire a large number of sample values [1] of $\widehat{\text{CTR}}(t)$ under $\mathcal{H}_0$, we can set $\tau$ as follows.

$$\tau \triangleq \sup\{x : \text{The fraction of } \widehat{\text{CTR}}(t) \text{ with } \widehat{\text{CTR}}(t) \leq x \text{ is less than or equal to } \kappa.\} \quad (8)$$

where supremum is taken to maximize the threshold and, in turn, minimize the miss detection probability.

## 3.6 Performance Analysis: Simulations

This section presents the simulation results of MCTRD-TS using Poisson traffic. Table 2 contains the explanation about simulation parameters. In simulations, Poisson processes are used for transmission processes of nodes, and the transmission delay is uniformly distributed in $[0, \Delta]$.

Clock skew uncertainties are represented by independent and identically distributed random variables $U_1, U_2, \ldots, U_n$, uniformly distributed in $[0, \alpha]$. We add $U_i$ to every timestamp of the $i$th node to emulate the effects of clock skews.

Fig. 6 contains receiver operating characteristics (ROCs) of MCTRD-TS with different number of observations[2]. When the number of observations increases, the ROC moves closer to the upper left corner (*i.e.*, zero error probabilities) as expected from theorem 1.

## 3.7 Performance Analysis: Experimental Results

This section presents the experimental results for MCTRD-TS in a network testbed.

[1]Sample values can be collected by applying MCTRD-TS to many sets of $\mathcal{H}_0$ traces. For example, assume that we have a normal traffic covering a sufficiently long time interval. Then, we can synthesize $\mathcal{H}_0$ traffic, based on the approximation that traces from disjoint time intervals are independent. If we are not able to acquire such traffic, we can instead use a good synthetic traffic model (*e.g.*, renewal process with heavy tail interarrival time).

[2]100 packets per node means that MCTRD-TS uses 100 packets per node for each detection trial.

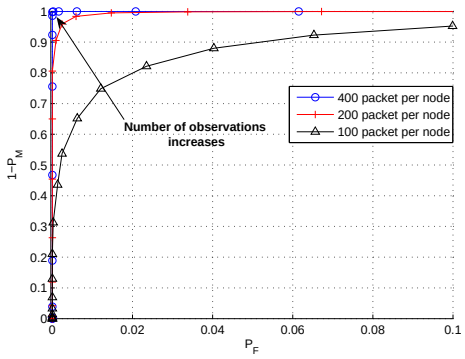



Figure 6: ROCs of MCTRD-TS with different number of observations: $n = 6$, $\lambda = 14$, $\Delta = 0.5$, $f_c = 0.2$, $\alpha = 0.1$, 10,000 Monte Carlo runs.

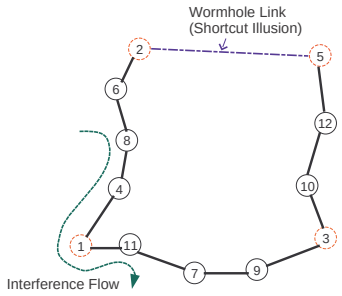



Figure 7: Test Topology

### 3.7.1 Test Environment

We evaluated MCTRD-TS for wormhole tunnel localization accuracy by using it to process data generated in a network testbed at the Army Research Laboratory (ARL). The testbed is based on Naval Research Laboratory's Mobile Ad-hoc Network Emulator (MANE) [1]. A MANE system consists of a collection of Linux-based test nodes and one or more emulation servers that are interconnected via Ethernet. These systems are logically arranged in a hub and spoke configuration such that all traffic between test nodes must be relayed by an emulation server. The emulation servers model the geographic positioning and movement of test nodes, and determine whether packets sent between them should be relayed transparently or dropped as a function of emulated distance, transmission power, noise, and other factors.

Our experiment used 12 test nodes equipped with the Fedora Core 3 operating system and the OLSR ad hoc routing daemon supported by olsr.org [2]. The MANE testbed was configured to position these nodes in the U-shaped topology depicted in Fig. 7. Here, the path (2, 6, ..., 12, 5) can be interpreted as the tunnel path estimate given by the Path Estimation block of the localization system. Under $\mathcal{H}_1$, nodes 2, 1, 3, and 5 were configured to act as wormhole attackers. Nodes 2 and 5 were used as tunnel endpoints, with nodes 1 and 3 acting as relays, as explained in Section 2.1.

Each of these nodes runs a wormhole attack tool that uses the *vtun* utility [4] to create wormhole tunnels. Tunnels may be configured to use either unreliable (UDP) or reliable (TCP) transport layers. Because the cumulative effect of packet loss over long tunnel path can prevent a wormhole link from stabilizing, making the attack ineffective, our tests used TCP-based tunnels.

In addition to the OLSR protocol messages sent between

Table 3: Packet Loss Probability

| Link | Prob | Link | Prob | Link | Prob |
|---|---|---|---|---|---|
| 2 - 6 | 0.0006 | 1 - 11 | 0.0021 | 3 - 10 | 0.0526 |
| 6 - 8 | 0.0012 | 11 - 7 | 0.0291 | 10 - 12 | 0.0055 |
| 8 - 4 | 0.0007 | 7 - 9 | 0.0033 | 12 - 5 | 0.0433 |
| 4 - 1 | 0.0318 | 9 - 3 | 0.0193 | | |

nodes 2 and 5, we created a flow of synthetic application traffic between these nodes. Both kinds of traffic are covertly forwarded by the attackers through the wormhole tunnel. To create this traffic, we used NRL's Real-Time Application Representative (RAPR) [3]. We configured RAPR to generate a bursty flow of UDP packets resembling voice over IP (VoIP) traffic.

### 3.7.2 Results and Analysis

We evaluated MCTRD-TS using a self-contained in-band wormhole. However, we anticipate that MCTRD-TS would exhibit similar performance for a tunnel used in an extended in-band wormhole.

The objective of MCTRD-TS is to detect the presence of an information flow in the eleven-hop path (2, 6, . . . , 12, 5).

The experimental setting for each hypothesis is as follows. Under $\mathcal{H}_0$, each node transmits VoIP packets independently from other nodes. Under $\mathcal{H}_1$, node 2, node 1, node 3, and node 5 are attackers, and they form the eleven-hop in-band wormhole tunnel described above. Furthermore, as illustrated in Fig. 7, a VoIP interference flow having the same rate as the tunnel flow is injected on the path (8, 4, 1, 11) thereby making the experiment more realistic. Under $\mathcal{H}_1$, TCP tunnels are created between node 2 and node 1, between node 1 and node 3, and between node 3 and node 5. As explained in Section 2.1, when the intermediate attackers, node 1 and node 3, receive packets from one TCP tunnel and send them through the next TCP tunnel, de-encapsulation and re-encapsulation occur. It was occasionally observed that, during this process, two or more TCP packets merge into a bigger TCP packet. The effect is similar to introducing chaff packets.

Observations for detection consist of the timestamps of TCP/UDP data packets and OLSR control packets. From every node except node 5, we gather the transmission timings of every packet with a non-zero length payload, whose next hop address includes the node's next hop in the suspect path. From node 5, we collect the timings of received packets with a non-zero length payload.

In our experiment, the link connectivity is modeled by the Free Space Path Loss (FSPL) propagation model. In this setting, every link has a certain packet loss probability. The packet loss probabilities of one-hop links in the suspect path are given in Table 3. Note that lost packets will also act as chaff and, furthermore, will trigger TCP retransmissions.

Fig. 8 is the plot of $\widehat{\text{CTR}}$s calculated by MCTRD-TS, under $\mathcal{H}_0$ and $\mathcal{H}_1$ respectively, using 1,000 packets per node. The $\widehat{\text{CTR}}$ value of index $i$ represents the $\widehat{\text{CTR}}$ calculated using the $i$th set of data consisting of 1,000 packets per node. The uppermost plot is $\widehat{\text{CTR}}$ values under $\mathcal{H}_0$, the bottom one is $\widehat{\text{CTR}}$ values under $\mathcal{H}_1$, and the middle straight line represents a proper threshold. We can observe that despite the packet loss and the presence of large amount of chaff noise, two hypotheses are quite separable. When more than 2,000 observations per node were used, $\widehat{\text{CTR}}$s for

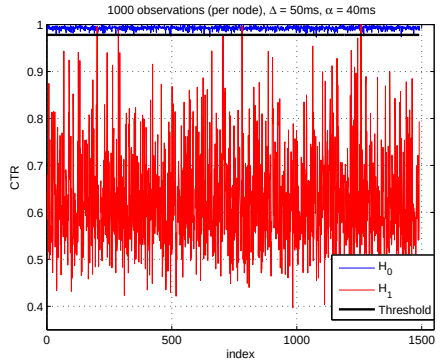



Figure 8: $\widehat{\text{CTR}}$ plots of MCTRD-TS: $\mathcal{H}_0$ rate = 19.4 packet/sec, $\mathcal{H}_1$ rate = 18.2 packet/sec, $\Delta = 50ms$, $\alpha = 40ms$, and the number of observation is 1,000 packets per node.

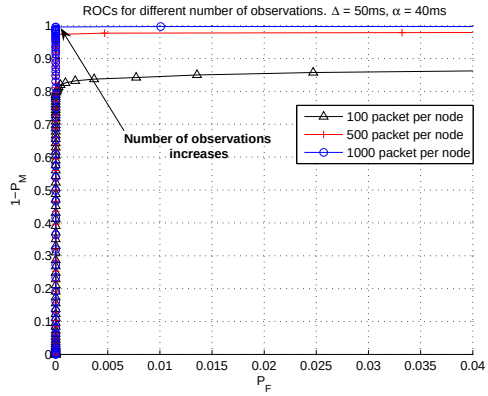



Figure 9: ROCs of MCTRD-TS with different number of observations: $\Delta = 50ms$, $\alpha = 40ms$. 1,500 Monte Carlo runs for 1,000 packet/node case, 3,000 Monte Carlo runs for 500 packet/node case, and 15,000 Monte Carlo runs for 100 packet/node case.

two hypotheses were completely separated. When we ran MCTRD-TS over independent Poisson traffic with the rate 19.4 packet per sec, $\widehat{\text{CTR}}$s stayed within [0.75, 0.82] range, which is much lower than $\mathcal{H}_0$ $\widehat{\text{CTR}}$ values from the synthetic VoIP traffic. This implies that $\limsup_{t\to\infty} \widehat{\text{CTR}}$ under $\mathcal{H}_0$ is much lower when the transmission processes are Poisson processes. From (6) and (7), we can infer that the detector is more robust to chaff when used over synthetic VoIP traffic than Poisson traffic. This argument agrees with the claim in [15] that the Poisson assumption provides the lower bound on the actual detection performance.

Fig. 9 contains ROCs of MCTRD-TS with different numbers of observations. ROCs are achieved by varying the threshold of MCTRD-TS from 0 to 1 while computing the false alarm probability and miss detection probability for each threshold. The comparison of ROCs shows that a larger number of observations result in better detection performance.

Table 4 shows examples of setting threshold $\tau$, and the resulting error probabilities. We employed (8) in section 3.5 with $\kappa = 0.004$. There exist errors due to the lack of sample $\mathcal{H}_0$ $\widehat{\text{CTR}}$ values. From the table, we can see the clear trade-off between the observation time and detection performance (also observable in Fig. 9). Note that the observation time depends on the tunnel stability which is affected by the node mobility. Thus, we can infer how the mobility of nodes may

Table 4: Error Probability versus the number of observations

| number of observations per node | $\tau$ | $P_F$ | $P_M$ |
|---|---|---|---|
| 100 | 0.935 | 0.006 | 0.161 |
| 500 | 0.970 | 0.005 | 0.026 |
| 1000 | 0.978 | 0.007 | 0.003 |

affect detection performance, noting that tunnel instability may also degrade the effectiveness of the attack.

# 4. TUNNEL PATH ESTIMATION

In this section, we present the tunnel path estimation algorithm that is used in the Path Estimation block of the localization system.

## 4.1 Tunnel Path Estimation Problem

Let $G = (\mathcal{N}, \mathcal{A})$ be a directed graph representing the MANET topology, and assume that an in-band wormhole attack exists. Let $N \triangleq |\mathcal{N}|$, and $R_i$ $(i = 1, \ldots, N)$ denote the nodes, where $R_1$ and $R_N$ are the tunnel endpoints. $(i, j)$ is in $\mathcal{A}$ if and only if $R_i$ can send packets directly to $R_j$. By observing $(\mathbf{S}_i)_{i=1}^N$ for some time $t$ $(t > 0)$, the goal is to find the true tunnel path (*i.e.*, the path containing an information flow) among all possible paths from $R_1$ to $R_N$.

Note that the above formulation assumes that we start with the correct tunnel endpoints. In practice, the Attack Alarm block can produce false alarms or identify the wrong tunnel endpoints. Even in that case, our path estimation algorithm will still select the most likely path. However, the path will be proved innocent in the Validation block with high probability. Hence, we focus on the case in which the decision of the Attack Alarm block is correct. Although the clock skew problems can be resolved as in Section 3.5, for simplicity, we assume that node clocks are synchronized.

## 4.2 Incremental Optimal Scheduling

Before presenting the tunnel path estimation algorithm, we introduce a new minimum-CTR calculation method, which is used as a building block of the path estimation algorithm. Finding the minimum CTR is equivalent to finding a maximum number of relays. Here, we formally define a *relay* as a sequence of timings $(t_i)_{i=1}^n$, $t_i \in \mathcal{S}_i$, satisfying $t_i \in [t_{i-1}, t_{i-1} + \Delta]$, $2 \le i \le n$ (*i.e.*, satisfying causality and the delay bound). Relays $(a_i)_{i=1}^n$ and $(b_i)_{i=1}^n$ are said to be *disjoint* if $a_i \neq b_i, \forall i$. And, a collection of disjoint relays is said to be *order-preserving* if for any two relays $(a_i)_{i=1}^n$, $(b_i)_{i=1}^n$, $a_1 < b_1$ implies $a_i < b_i$, $2 \le i \le n$.

In [15], given the realization of transmission processes, MBDR is shown to find a maximum number of disjoint relays by finding the earliest [3] order-preserving relays. However, if we run MBDR for a large number of paths, it becomes inefficient in that it cannot reuse the calculation on the shared paths. For instance, assume that we want to find a maximum number of disjoint relays for $(\mathbf{s}_i)_{i=1}^n$ and $(\mathbf{s}_i)_{i=1}^{n+1}$. Then, it is natural to expect that there would be a way to utilize the calculation on $(\mathbf{s}_i)_{i=1}^n$ for the calculation on $(\mathbf{s}_i)_{i=1}^{n+1}$. However, in case of MBDR, the calculation on $(\mathbf{s}_i)_{i=1}^{n+1}$ cannot benefit from the calculation on $(\mathbf{s}_i)_{i=1}^n$ due to its recursive characteristic. To improve this drawback, we propose a matching algorithm, called Incremental Optimal Scheduling (IOS), which calculates the CTRs of increasing paths while benefiting from the previous calculations.

Given the realizations $(\mathbf{s}_i)_{i=1}^n$, IOS finds a maximum number of disjoint relays for each $(\mathbf{s}_i)_{i=1}^k$, $2 \le k \le n$, as follows:

1. Set $L(i, 1) = \{s_1(i)\}$, $1 \le i \le |\mathcal{S}_1|$, and $k = 2$.
2. Define $L(i, k)$ to be the set of epochs in $\mathcal{S}_k$ which can be matched[4] to at least one element in $L(i, k-1)$.
3. Find the earliest order-preserving relays for $(\mathbf{s}_i)_{i=1}^k$; first, find the earliest relay containing $s_1(1)$, then find the earliest order-preserving relay containing $s_1(2)$, and repeat this until we reach the last epoch of $\mathcal{S}_1$. Based on the sets $L(i, j)$, $1 \le i \le |\mathcal{S}_1|$, $1 \le j \le k$, this can be done by finding minimums of sets (refer to lines 10-29 in Table. 5 in Appendix A).
4. After the matching is finished, calculate $\widehat{\text{CTR}}$ for $(\mathbf{s}_i)_{i=1}^k$. If the timing $s_1(i)$ is contained in one of the found relays, then remove the epochs in $L(i, j)$, $1 \le j \le k$, which are earlier than the relay; otherwise, make $L(i, j)$, $1 \le j \le k$, empty.
5. If $k = n$, terminate; otherwise, $k \leftarrow k + 1$ and go to 2.

After the iteration for $k = m$ is finished, $L(i, j)$, $1 \le i \le |\mathcal{S}_1|$, $1 \le j \le m$, consists of the epochs $t \in \mathcal{S}_j$ which can possibly be an entry of the IOS relay containing $s_1(i)$, in the later iterations. In other words, if $t \in \mathcal{S}_j$ is not in $L(i, j)$, then $t$ can never be in the IOS relay containing $s_1(i)$ in the later iterations. In step 4, epochs which no longer have such possibility are removed from the sets.

IOS attempts to find the earliest order-preserving relays, which are the same as what MBDR finds [5]. The rationale behind the above paragraph and step 4 is based on two characteristics of MBDR : (i) if $s_1(j)$ is not contained in any relay found by MBDR over $(\mathbf{s}_i)_{i=1}^{m-1}$, then it is not contained in any relay found by MBDR over $(\mathbf{s}_i)_{i=1}^m$; (ii) if MBDR on $(\mathbf{s}_i)_{i=1}^{m-1}$ finds a relay $(a_i)_{i=1}^{m-1}$ and MBDR on $(\mathbf{s}_i)_{i=1}^m$ finds a relay $(b_i)_{i=1}^m$, where $a_1 = b_1$, then $a_i \le b_i$, $1 \le i \le m-1$. The implementation of IOS is given in Table 5 in Appendix A. The following theorem states the optimality of IOS.

*Theorem 2.* For any realization $(\mathbf{s}_i)_{i=1}^n$, IOS finds a maximum number of disjoint relays.

*Sketch of Proof:* See *Appendix B* ■

If we use IOS to find a maximum number of relays for $(\mathbf{s}_i)_{i=1}^n$ and $(\mathbf{s}_i)_{i=1}^{n+1}$, the calculation for $(\mathbf{s}_i)_{i=1}^{n+1}$ can be effectively reduced by using the sets $L(i, j)$, $1 \le i \le |\mathcal{S}_1|$, $1 \le j \le n$, resulting from the calculation on $(\mathbf{s}_i)_{i=1}^n$. We denote such calculation by $\widehat{\text{IOS}}(T, L, \mathbf{s}_{n+1}) = (\tilde{T}, \tilde{L}, \widetilde{\text{CTR}})$ , where $T$ and $\tilde{T}$ are the number of all epochs in $(\mathbf{s}_i)_{i=1}^n$ and $(\mathbf{s}_i)_{i=1}^{n+1}$ respectively, $\tilde{L}(i, j)$, $1 \le i \le |\mathcal{S}_1|$, $1 \le j \le n+1$, are new resulting sets, and $\widetilde{\text{CTR}}$ is $\widehat{\text{CTR}}$ calculated for $(\mathbf{s}_i)_{i=1}^{n+1}$. The complexity of $\widehat{\text{IOS}}(T, L, \mathbf{s}_{n+1})$ is $O(|\mathcal{S}_1|(n \log n))$ (see Appendix A.). However, if we use MBDR, since it does not benefit from the previous calculation on $(\mathbf{s}_i)_{i=1}^n$, the complexity of calculation for $(\mathbf{s}_i)_{i=1}^{n+1}$ is $O(|\mathcal{S}_1|n^2)$ [15].

[3] Given two relays $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$, we say that $(a_i)_{i=1}^n$ is *earlier* than $(b_i)_{i=1}^n$ if $\exists m \ge 1$ s.t., $a_i \le b_i$ for $1 \le i < m$, and $a_m < b_m$.

[4] $a \in \mathcal{S}_{i+1}$ can be matched to $b \in \mathcal{S}_i$ iff $a \in [b, b + \Delta]$.

[5] It is shown in the proof of theorem 2.

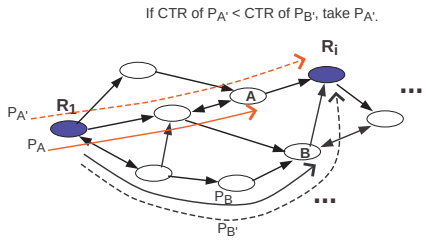



Figure 10: At each iteration, $R_i$ looks for neighbors which have an outgoing arc to $R_i$ (Here, $A$ and $B$.). $P_A$ and $P_B$ are survivor paths of $A$ and $B$ calculated in the last iteration, and $P_{A'}$ and $P_{B'}$ are their one-hop extensions toward $R_i$. If $\widehat{\text{CTR}}$ of $P_{A'}$ is lower than that of $P_{B'}$, then $R_i$ sets its survivor path to be $P_{A'}$.

## 4.3 Minimum-CTR Tunnel Path Estimation

Using $\widehat{\text{IOS}}$ as a building block, we propose a tunnel path estimation algorithm, called Minimum-CTR Tunnel Path Estimation (MCTR-PE). The main idea is that every node saves one survivor path having itself as the end vertex and $R_1$ as the start vertex. At each iteration, $R_i$ sets its survivor path to be the best path among one-hop extensions of its neighbors' survivor paths (extended by adding $R_i$ as the end vertex). The path with the minimum $\widehat{\text{CTR}}$ is regarded as the best path, where $\widehat{\text{CTR}}$ for each extension is calculated by $\widehat{\text{IOS}}$. After $N$ iterations, MCTR-PE returns the survivor path of $R_N$. Fig. 10 illustrates how a node sets its survivor path at each iteration. The rationale behind MCTR-PE is that the path with lower $\widehat{\text{CTR}}$ more tends to contain an information flow (*i.e.*, more tends to be a true tunnel path). Given $(\mathbf{s}_i)_{i=1}^N$, MCTR-PE works as follows:

1. Let $h = 1$. For $i = 2, \ldots, N$, let $I_i = \{j \in \mathcal{N} | (j, i) \in \mathcal{A}\}$. $R_i$ saves one survivor path $p_i$. Initially, $p_1 = (1)$, and $p_i = ()$, $i \neq 1$.
2. For $i = 2, \ldots, N$, save $p_i$ into $\hat{p}_i$. And, for $i = 2, \ldots, N$, let $C_i = \{j \in I_i | \, i \notin \hat{p}_j \text{ and } 1 \in \hat{p}_j\}$.
3. For $i = 2, \ldots, N$, if $C_i$ is not empty, make one-hop extension of each $\hat{p}_j$, $j \in C_i$, by adding $i$ as the end vertex. Among the extended paths, pick the path with the minimum $\widehat{\text{CTR}}$ found by $\widehat{\text{IOS}}$. Save the selected path into $p_i$.
4. Increase $h$ by 1. If $h < N$, go to the step 2; otherwise, return the survivor path of $R_N$.

The implementation of MCTR-PE is given in table 6 in Appendix A, and the complexity [6] is $O(|\mathcal{S}_1||\mathcal{A}|N^2 \log N)$.

## 4.4 Performance Analysis

We tested MCTR-PE simulating the network topology shown in Fig. 11, where the path denoted by the arrow is the in-band wormhole tunnel path. There are 60 possible paths from $R_1$ to $R_{19}$. We set every node to transmit at the same rate (4 packets per sec), and the delay constraint is 0.5 sec. A transmission process of a node not on the tunnel path is independent of all other nodes. Error detection probability [7] versus the flow strength is plotted in Fig. 12. The

[6] Total $N-1$ iterations are excuted, and in each iteration $\widehat{\text{IOS}}$ is excuted at most $|\mathcal{A}|$ times.

[7] Error detection probability is the probability that MCTR-PE chooses a wrong path.

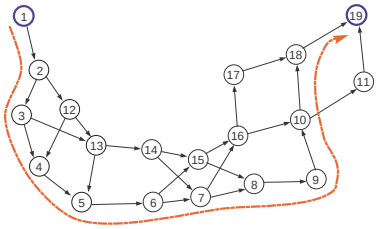



Figure 11: Test Topology for MCTR-PE

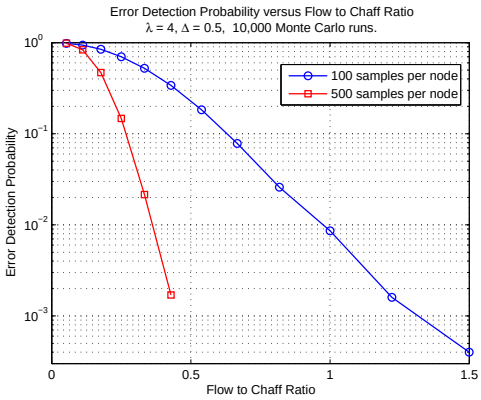



Figure 12: MCTR-PE Results: Error Probability versus FCR. For 500 samples/node case, no error occurred for FCR $> 0.43$. For 100 samples/node case, no error occurred for FCR $> 1.5$.

ratio $\frac{|\{\text{flow packets}\}|}{|\{\text{chaff noise}\}|}$ on the tunnel path, denoted by *Flow to Chaff Ratio (FCR)*, is used as the metric to characterize the flow strength. The error detection propability shows an exponential decay as FCR increases, and when 500 samples per node are used, it shows reasonably low error probability even when the flow strength is weak (FCR $< 0.5$). In addition, the increase in the number of observations leads to a significant decrease in the error detection probability.

# 5. DISCUSSION

While the presented results are encouraging, verifying that the MCTRD-TS and MCTR-PE algorithms can identify wormhole tunnels accurately under more realistic conditions will require additional research. In particular, MCTR-PE needs to be tested with more realistic traces than Poisson traffic. Both algorithms should be tested in larger topologies, with more complex background traffic. In practice, a suspected path may partially overlap with many other flow paths. Thus, the transmission activities of groups of nodes along the path may be correlated, even when the suspected path is innocent. In such situations, the detection of the tunneled traffic becomes more difficult, and attaining the detection accuracy in Section 3.7 and Section 4.4 will likely require increasing the number of observations per trial.

As noted earlier, we have assumed that the tunnel path of a persistent wormhole attack will remain stable for at least one period of sufficient duration to allow logging the required number of packet transmissions, e.g., 100-1000 packets. While this assumption appears to be a reasonable one in general, its validity depends on the mobility of the nodes. Regardless, the performance of MCTRD-TS and MCTR-PE should be evaluated in the presence of network mobility.

## 6. CONCLUSION

This paper presents timing-based algorithms for localizing in-band wormhole tunnels in MANETs, and proposes a conceptual model for a tunnel localization system that combines our algorithms with existing techniques for detecting the presence of a wormhole attack. We believe these are the first algorithms directed at identifying such tunnels in their entirety, including colluding relay nodes. We have described their mathematical basis, and presented performance evaluations using Poisson traffic and data from a MANET emulation testbed that included synthetic VoIP traffic and an implementation of a wormhole attack. Simulation and experimental results indicate that the algorithms exhibit high accuracy given an opportunity to obtain a sufficient number of packet observations, and are robust to probabilistic packet loss, chaff, and clock skew uncertainty, which are key characteristics of MANET environments.

## 7. REFERENCES


[1] Mobile Ad-hoc Network Emulator (MANE): http://cs.itd.nrl.navy.mil/work/mane/index.php.

[2] OLSR.org: http://www.olsr.org.

[3] RAPR - The Real-Time Application Representative: http://cs.itd.nrl.navy.mil/work/rapr/index.php.

[4] Virtual Tunnels over TCP/IP Networks: http://vtun.sourceforge.net.

[5] C. Adjih, T. Clausen, A. Laouiti, P. Muhlethaler, and D. Raffo. Securing the OLSR routing protocol with or without compromised nodes in the network. Technical Report ISRN INRIAR/RR-5494, INRIA, Feb. 2005.

[6] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens. Mitigating Byzantine Attacks in Ad Hoc Wireless Networks. Technical Report Version 1, Department of Computer Science, Johns Hopkins University, Mar. 2004.

[7] B. Awerbuch, R. Curtmola, D. Holmer, H. Rubens, and C. Nita-Rotaru. On the Survivability of Routing Protocols in Ad Hoc Wireless Networks. In *1st International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2005)*, Sept. 2005.

[8] A. Blum, D. Song, and S. Venkataraman. Detection of Interactive Stepping Stones: Algorithms and Confidence Bounds. In *Conference of Recent Advance in Intrusion Detection (RAID)*, Sophia Antipolis, French Riviera, France, September 2004.

[9] L. Buttyán, L. Dóra, and I. Vajda. Statistical Wormhole Detection in Sensor Networks. In *Second European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS 2005)*, Visegrád, Hungary, July 2005.

[10] T. Clausen and P. Jacquet. Optimized Link State Routing (OLSR): http://www.ietf.org/rfc/rfc3626.txt, Oct. 2003.

[11] D. Donoho, A. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford. Multiscale stepping-stone detection: Detecting pairs of jittered interactive streams by exploiting maximum tolerable delay. In *5th International Symposium on Recent Advances in Intrusion Detection, Lecture Notes in Computer Science 2516*, 2002.

[12] N. Ferguson and B. Schneier. *Practical Cryptography*. John Wiley & Sons, Inc., Indianapolis,IN, 2003.

[13] M. Gorlatova, P. Mason, M. Wang, L. Lamont, and R. Liscano. Detecting Wormhole Attacks in Mobile Ad Hoc Networks through Protocol Breaking and Packet Timing Analysis. In *MILCOM 2006*, Washington DC, Oct. 2006.

[14] T. He and L. Tong. Detecting Encrypted Stepping-Stone Connections. *IEEE Transactions on Signal Processing*, 55(5):1612–1623, May 2007.

[15] T. He and L. Tong. Detection of Information Flows. *IEEE Trans. Inf. Theory*, 54:4925–4945, Nov. 2008.

[16] T. He and L. Tong. Distributed Detection of Information Flows. *IEEE Trans. Inf. Forensics Security*, 3:390–403, Sept. 2008.

[17] Y. Hu, A. Perrig, and D. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In *Proceedings of IEEE Infocom 2003*, San Francisco, CA, Apr. 2003.

[18] I. Khalil, S. Bagchi, and B. Shroff. LITEWORP: A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, Yokohama, Japan, June 2005.

[19] P. Kruus, D. Sterne, R. Gopaul, M. Heyman, B. Rivera, P. Budulas, B. Luu, T. Johnson, N. Ivanic, and G. Lawler. In-Band Wormholes and Countermeasures in OLSR Networks. In *2nd International Conference on Security and Privacy in Communication Networks (SecureComm 2006)*, Baltimore, MD, Aug. 2006.

[20] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. W. Chang. Preventing Wormhole Attacks on Wireless Ad Hoc Networks: A Graph Theoretic Approach. In *IEEE Wireless Communications and Networking Conference (WCNC) 2005*, New Orleans, LA, Mar. 2005.

[21] J. Shao. *Mathematical Statistics*. Springer, 2003.

[22] D. Sterne, R. Gopaul, G. Lawler, P. Kruus, B. Rivera, and K. Marcus. Countering False Accusations and Collusion in the Detection of In-Band Wormholes. In *Annual Computer Security Applications Conference*, Miami Beach, Florida, Dec. 2007.

[23] X. Wang and D. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter-packet delays. In *Proc. of the 2003 ACM Conference on Computer and Communications Security*, pages 20–29, 2003.

[24] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad Hoc Networks. In *Proceedings of The Sixth International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, Aug. 2000.

[25] S. Zheng, T. Jiang, J. S. Baras, A. Sonalker, D. Sterne, R. Gopaul, and R. Hardy. Intrusion Detection of In-Band Wormholes in MANETs using Advanced Statistical Methods. In *MILCOM 2008*, San Diego, CA, Nov. 2008.

Table 5: Incremental Optimal Scheduling (IOS)

IOS($\mathbf{s}_1, \ldots, \mathbf{s}_n, \Delta, t$):

1: for $i = 1:1:|\mathcal{S}_1|$, $L(i,1) = \{S_1(i)\}$. end.
2: for $j = 1:1:n-1$, $\mathrm{CTR}(j) \leftarrow 0$. end.
3: $T \leftarrow |\mathcal{S}_1|$, $k \leftarrow 2$.
4: While $k \le n$
5: $T \leftarrow T + |\mathcal{S}_k|$.
6: for $i = 1:1:|\mathcal{S}_1|$
7: $L(i,k) \leftarrow \{x \in \mathcal{S}_k : [x-\Delta, x] \cap L(i,k-1) \neq \emptyset\}$
8: end
9: for $j = 1:1:k$, $I(j) \leftarrow 0$, $J(j) \leftarrow 0$. end. $f \leftarrow 0$.
10: for $i_1 = 1:1:|\mathcal{S}_1|$
11: $\sigma \leftarrow 0$, $u_1 \leftarrow 0$.
12: for $j = 2:1:k$
13: $\hat{L}_j \leftarrow \{x \in L(i_1, j) : x > J(j) \text{ and } x \ge u_{j-1}\}$
14: If $\hat{L}_j$ is empty, $\sigma \leftarrow 1$ and break. end
15: $u_j \leftarrow \min \hat{L}_j$.
16: end
17: If $\sigma = 0$
18: $I(k) \leftarrow u_k$.
19: for $j = k-1:-1:2$
20: $I(j) \leftarrow \min(\hat{L}_j \cap [I(j+1) - \Delta, I(j+1)])$.
21: end
22: for $j = 2:1:k$
23: $L(i_1, j) \leftarrow \{x \in L(i_1, j) : x \ge I(j)\}$.
24: end
25: $J \leftarrow I$, $f \leftarrow f + 1$.
26: else
27: for $j = 1:1:k$, $L(i_1, j) \leftarrow \emptyset$. end.
28: end
29: end
30: $\mathrm{CTR}(k-1) \leftarrow \frac{T-kf}{T}$. $k \leftarrow k+1$.
31: end
32: return CTR

Table 6: Minimum-CTR Tunnel Path Estimation (MCTR-PE)

MCTR-PE($\mathbf{s}_1, \ldots, \mathbf{s}_N, \Delta, t$):

1: $p_1 \leftarrow (1)$. $T_1 \leftarrow |\mathcal{S}_1|$
2: $L_1$: $|\mathcal{S}_1| \times 1$ array, $L_1(i,1) = \{s_1(i)\}$, $1 \le i \le |\mathcal{S}_1|$.
3: for $i = 2:1:N$
4: $p_i \leftarrow ()$, $T_1 \leftarrow 0$ $I_i \leftarrow \{j \in \mathcal{N} | (j,i) \in \mathcal{A}\}$
5: $L_i$: $|\mathcal{S}_1| \times 1$ array, $L_i(j,1) = \emptyset$, $1 \le j \le |\mathcal{S}_1|$.
6: end
7: $h = 1$.
8: While $h < N$
9: for $i = 2:1:N$
10: $\hat{p}_i \leftarrow p_i$, $\hat{T}_i \leftarrow T_i$
11: $\hat{L}_i \leftarrow L_i$
12: $C_i \leftarrow \{j \in I_i | \, i \notin \hat{p}_j \text{ and } 1 \in \hat{p}_j\}$.
13: end
14: for $i = 2:1:N$
15: if $C_i \neq \emptyset$
16: for all $j \in C_i$
17: $(\tilde{T}_j, \tilde{L}_j, \widetilde{\mathrm{CTR}}_j) \leftarrow \widehat{\mathrm{IOS}}(\hat{T}_j, \hat{L}_j, \mathbf{s}_i)$.
18: end
19: $j^* \leftarrow \arg\min_{j \in C_i} \widetilde{\mathrm{CTR}}_j$
20: $p_i \leftarrow \mathrm{extend}(\hat{p}_{j^*}, i)$.
21: $T_i \leftarrow \tilde{T}_{j^*}$, $L_i \leftarrow \tilde{L}_{j^*}$.
22: end
23: end
24: $h \leftarrow h + 1$
25: end
26: return $p_N$

*extend($\hat{p}_{j^*}, i$): 1-hop extension of $\hat{p}_{j^*}$, where $i$ is added at its end.

# APPENDIX

## A. IMPLEMENTATIONS

The implementations of IOS and MCTR-PE are given in Table 5 and Table 6 respectively. $\widehat{\mathrm{IOS}}(T, L, \mathbf{s}_{n+1})$ executes lines 5-30 in Table 5 once for $k = n+1$. Let $\lambda$ be the maximum among the rates of $\mathbf{s}_1, \ldots, \mathbf{s}_{n+1}$, and assume that the measurements are ordered. The main steps of $\widehat{\mathrm{IOS}}$ are lines 13, 20, and 23. Since $|\hat{L}_j| \le n\lambda\Delta$ on average, $1 \le j \le n+1$, a single execution of three lines takes $O(\log n)$. Hence, the complexity of $\widehat{\mathrm{IOS}}(T, L, \mathbf{s}_{n+1})$ is $O(|\mathcal{S}_1|(n \log n))$.

## B. PROOF OF THEOREM 2

MBDR in [15] finds the earliest order-preserving relays, and it was proved to find a maximum number of disjoint relays. Let $(T_i^n(k))_{k=1}^n$ be the $i$th relay found by IOS over $(\mathbf{s}_i)_{i=1}^n$, and $(\hat{T}_i^n(k))_{k=1}^n$ the $i$th relay found by MBDR. We will show that $(T_i^n(k))_{k=1}^n = (\hat{T}_i^n(k))_{k=1}^n$, $\forall i$, $n$.

We use mathematical induction. When $n = 2$, it is easy to check that $(T_i^2(k))_{k=1}^2 = (\hat{T}_i^2(k))_{k=1}^2$, $\forall i$. Assume that $(T_i^n(k))_{k=1}^n = (\hat{T}_i^n(k))_{k=1}^n$, $\forall i$, is true for $n \le m-1$. Then, showing $(T_i^m(k))_{k=1}^m = (\hat{T}_i^m(k))_{k=1}^m$, $\forall i$, concludes the proof. The proof for $(T_1^m(k))_{k=1}^m = (\hat{T}_1^m(k))_{k=1}^m$ is given below. For $i \ge 2$, it can be proved in the same manner by using another induction (*i.e.*, assume the statement is true for $i \le b-1$, and prove that it is also true for $i = b$.).

Since MBDR finds the earliest order-preserving schedules, $\hat{T}_1^m(1) \le T_1^m(1)$. And, if $\hat{T}_1^m(2) > T_1^m(2)$, then

$$\hat{T}_1^m(1) \le T_1^m(1) \le T_1^m(2) < \hat{T}_1^m(2) \le \hat{T}_1^m(1) + \Delta$$

and thus $(\hat{T}_1^m(1), T_1^m(2), T_1^m(3), \ldots, T_1^m(m))$ is earlier than $(\hat{T}_i^m(k))_{k=1}^m$ contradicting the operation of MBDR. Hence, $\hat{T}_1^m(2) \le T_1^m(2)$, and similarly, $\hat{T}_1^m(k) \le T_1^m(k)$, $1 \le k \le m$. Next, we show $T_1^m(k) \le \hat{T}_1^m(k)$, $1 \le k \le m$.

When IOS finds the earliest relay containing $\hat{T}_1^m(1)$ (*i.e.*, runs lines 11-28 of Table. 5), $J(j) = 0$, $1 \le j \le m$, because $\hat{T}_1^m(1) \le T_1^m(1)$. Let $w$ be the index such that $s_1(w) = \hat{T}_1^m(1)$. The fact that $(\hat{T}_1^m(k))_{k=1}^m$ is a relay found by MBDR implies $\hat{T}_1^m(k) \in L(w,k)$, $1 \le k \le m$. This results from the induction hypothesis and two properties of MBDR: (i) if $s_1(w)$ is not contained in any relay found by MBDR over $(\mathbf{s}_i)_{i=1}^{m-1}$, then it is not contained in any relay found by MBDR over $(\mathbf{s}_i)_{i=1}^m$; (ii) if MBDR on $(\mathbf{s}_i)_{i=1}^{m-1}$ gives a relay $(a_i)_{i=1}^{m-1}$ and MBDR on $(\mathbf{s}_i)_{i=1}^m$ gives a relay $(b_i)_{i=1}^m$, where $a_1 = b_1$, then $a_i \le b_i$, $1 \le i \le m-1$.

Assume that timings are positive. $\hat{T}_1^m(k) \in L(w,k)$, $1 \le k \le m$, implies $u_1 = 0$, $u_2 = \min\{x \in L(w,2) : x \ge u_1\} \le \hat{T}_1^m(2), \ldots, u_m = \min\{x \in L(w,m) : x \ge u_{m-1}\} \le \hat{T}_1^m(m)$, because $u_i \le \hat{T}_1^m(i) \le \hat{T}_1^m(i+1)$, $1 \le i \le m-1$.

Since none of $L(w,k)$, $1 \le k \le m$ is empty, $I(m) = u_m \le \hat{T}_1^m(m)$, in line 18. And, in line 20,

$$I(m-1) = \min(\{x \in L(w,m-1) : x \ge u_{m-1}\} \cap [I(m) - \Delta, I(m)])$$

The set on the right side is nonempty, and if $u_{m-1} \in [I(m) - \Delta, I(m)]$, then $I(m-1) = u_{m-1}$; otherwise, $I(m-1) = \min(L(w,m-1) \cap [I(m) - \Delta, I(m)]) \le \hat{T}_1^m(m-1)$, because $\hat{T}_1^m(m-1) \in L(w,m-1)$ and $I(m) \le \hat{T}_1^m(m)$. In both cases, $I(m-1) \le \hat{T}_1^m(m-1)$, and similarly $I(k) \le \hat{T}_1^m(k)$, $\forall k$.

On the other hand, $(I(k))_{k=1}^m$ is the first relay found by ISO, meaning that $(T_1^m(k))_{k=1}^m = (I(k))_{k=1}^m$. Hence, $T_1^m(k) \le \hat{T}_1^m(k)$, $\forall k$. Therefore, $(T_1^m(k))_{k=1}^m = (\hat{T}_1^m(k))_{k=1}^m$. ■

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