Opportunistic Piggyback Marking for IP Traceback

Long Cheng, Dinil Mon Divakaran, Wee Yong Lim, Vrizlynn L. L. Thing
Cyber Security & Intelligence Department, Institute for Infocomm Research (I²R), Singapore

Abstract—IP traceback is a solution for attributing cyber attacks, and it is also useful for accounting user traffic and network diagnosis. Marking-based traceback (MBT) has been considered a promising traceback approach, and has received considerable attention. However, we find that the traceback message delivery problem in MBT, which is important to the successful completion of a traceback, has not been adequately studied in the literature. To address this issue, we present the design, analysis, and evaluation of opportunistic piggyback marking (OPM) for IP traceback in this work.

OPM distinguishes itself from existing works by decoupling the traceback message content encoding and delivery functions in MBT, and efficiently achieves expedited and robust traceback message delivery by exploiting piggyback marking opportunities. Based on the proposed OPM scheme, we then present the flexible marking-based traceback framework, which is a novel design paradigm for IP traceback and has several favorable features for practical deployment of IP traceback. Through numerical analysis and comprehensive simulation evaluations, we demonstrate that our design effectively reduces the traceback completion delay and router processing overhead, and increases the message delivery ratio compared with other baseline approaches.

Index Terms—IP Traceback; Marking-based Traceback; Opportunistic Piggyback Marking; Network Forensics

I. INTRODUCTION

Attacks on the Internet are a growing threat. For example, as of 2014, the frequency of globally recognized distributed denial-of-service (DDoS) attacks reached an average rate of 28 per hour [1]. One challenge in defending against DDoS attacks is that, source IP addresses are often spoofed by attackers in order to evade traceability and bypass access controls. IP traceback is a common solution to identify the sources of attack packets and the paths followed by these attack packets. It can mitigate the attack effects and enable forensic investigations of network attacks (e.g., isolating compromised hosts from the rest of the network, and holding the attackers accountable) [2]. Although IP traceback techniques are motivated by adversarial applications, they can also be used for a wide range of non-adversarial network analysis applications, such as traffic accounting, fault diagnosis, network bottleneck identification and path validation [3]–[5].

While a number of IP traceback techniques have been proposed in the literature [6], marking-based traceback (MBT) approach has received considerable attention. The basic idea of MBT is that routers convey their traceback messages (e.g., the identity information) to the end-hosts by marking on passing packets. Accordingly, an end-host can construct a graph of network paths traversed by these marked packets regardless of source IP address spoofing [7]. It is obvious that applying packet-level marking all the time on all traffic flows is unnecessary and it suffers the scalability problem which overloads routers by marking each passing packet [8]. In many proposed solutions [9]–[11], traceback mechanisms are activated in a reactive manner, e.g., triggered by attack detection systems when any abnormal traffic flow is detected.

Existing research efforts in MBT have been devoted to two key issues. The first issue is the traceback decision making at individual routers. When a router receives a packet, it makes a decision on whether to send a traceback message to the end-host or not. For example, in [12], only the edge ingress router closest to a packet’s source marks that packet. Whereas in [13], all routers on an attack path will send traceback messages to the end-host in a probabilistic manner. The second research issue is the message content encoding, which determines the information a router marks in the IP header. For example, a router may convey its full IP address [13], hash value of IP address [14], or huffman code [15] to the end-host. However, most existing MBT methods [4], [7], [8], [10], [13], [14], [16] either assume that IP header has enough space to hold traceback messages or suppose to trace all traffic flows, and thus take for granted that traceback messages are carried by packets being traced to the end-host, leaving the efficient message delivery an unaddressed problem.

There is a fundamental disconnect between the length of a traceback message and the marking space available in IP header. On one hand, authenticated and privacy-preserving marking is desirable in practical deployment, in order to prevent compromised routers from forging the markings of upstream routers [14] and alleviate ISP’s security concern on disclosing the network topology [17]. Providing authentication (e.g., routers digitally signing the markings) or privacy-preserving (e.g., hiding and dynamically changing router identities) features inevitably increase the bit-length of a traceback message. On the other hand, the marking space in IP header is rather limited. As a result, a message normally needs to be divided into multiple fragments to be delivered to the end-host [13], [18]. In addition, to trace the complete routing path of a packet or flow, all routers on the path are expected to send their respective traceback messages to the end-host. Due to these factors, existing MBT methods may take a long time to complete when flow rate of the traffic being traced is low, or even fail when tracing flows composed of only a few numbers of packets. These situations happen when tracing software exploit attacks [19], crime traffic such as gaining control of a computer system, or for network diagnosis purposes. Therefore, efficient delivery of traceback messages is an important problem in MBT.

Moreover, traditional MBT methods assume that traceback messages are transmitted to destinations of traffic flows being traced (e.g., victims). However, to enhance privacy preserving in traceback and relieve the burden of traceback information storage and path reconstruction at the end-hosts, routers may
send their traceback messages to one or more trusted servers in the network. In this case, exiting message delivery schemes cannot be applied since traceback messages can no longer be carried by packets being traced. This poses a significant challenge for efficient traceback message delivery in MBT.

To address the above challenges, in this work, we introduce the opportunistic piggyback marking, where traceback messages are conveyed in a store-and-mark manner. The main idea of our approach is to exploit free ride opportunities for fast and robust delivery of traceback messages to end-hosts. The contributions from this work are summarized as follows:

- To the best of our knowledge, this is the first work that explicitly addresses the message delivery problem in IP traceback. We first present the basic design of opportunistic piggyback marking (referred to OPM) for efficient delivery of traceback messages in MBT. To further improve the robustness of traceback, we then propose the advanced design of opportunistic piggyback marking, named as AOPM.
- We provide a theoretical analysis of OPM by modeling its process at each router using the $M^X/M/1/C$ finite queue with batch arrivals. We show that opportunistic piggyback marking can potentially achieve fast IP traceback, and reduce the chances of fragments being stored at intermediate routers for a long time.
- Based on the proposed OPM/AOPM, we present the flexible marking-based traceback (FMBT) framework, a novel design paradigm for IP traceback, which meets several favorable objectives that previous individual traceback schemes failed to satisfy simultaneously.
- Through extensive performance evaluation, we demonstrate that our design effectively reduces the traceback completion delay and router processing overhead, and increases the message delivery ratio compared with other baseline approaches. We also demonstrate the feasibility of FMBT framework by applying OPM/AOPM technique to deliver traceback messages.

The rest of the paper is organized as follows. Section II surveys the related work. Section III describes the preliminaries of this work. Section IV presents the basic design of opportunistic piggyback marking (OPM). Section V provides the theoretical analysis of MBT. Section VI elaborates the detailed description of our advanced design (AOPM). Section VII presents the flexible marking-based traceback (FMBT) framework. Simulation results are provided in Section VIII. Finally, Section IX concludes this work.

## II. RELATED WORK

Although many IP traceback methods have been proposed, the majority of research efforts over the past decade in this area can be broadly classified into three categories: marking-based, logging-based and hybrid approaches. We briefly survey the related works accordingly below.

### A. Marking-based Approaches

In marking-based traceback, routers embed identity information in the IP headers of passing packets to convey network path information to an end-host. Existing MBT methods can further be divided into Deterministic Packet Marking (DPM) [9], [12] and Probabilistic Packet Marking (PPM) [13]. Typically, DPM embeds the first ingress border router’s identity information on packets in a deterministic manner, while PPM probabilistically augments packets with partial path information as they traverse in the network. The goal of DPM is to locate the attack source, and the main purpose of PPM is to identify the attack path.

As a representative work in deterministic marking, Belenky et al. [12] proposed to store the source address (i.e., the first border router’s IP address) in the marking fields of passing packets. Although deterministic marking incurs less computational overhead to trace back to the attack source at the end-host side, it lacks incremental deployment property since it assumes that ingress routers are always traceback enabled. Moreover, it may overload the ingress routers by marking each passing packet compared with the probability-based measure. To reduce the number of marked packets, authors in [20] presented a flow-level deterministic marking method for traceback. More recently, Yu et al. [11] proposed a marking on demand (MOD) scheme based on the DPM mechanism to dynamically distribute router IDs in both temporal and space dimensions.

One of the pioneering probabilistic marking solutions was proposed by Savage et al. [13], which probabilistically marks packets with router’s identity information as they traverse routers through the Internet. Later on, different variants of PPM [7], [10], [14], [16], [21] have been proposed to improve the scalability and efficiency of probabilistic marking. Adler [22] revealed that an inherent tradeoff exists in PPM between the number of header bits used and the number of packets required to reconstruct the attack path. PPM based approaches are able to reconstruct the attack path only after receiving sufficient marked packets at the end-host, and may generate false positives [23]. Dong et al. [3] presented a comparative summary of different PPM schemes. For the PPM approach proposed by Savage et al. [13], more than 2500 packets are required to convey network path information to the destination. Other methods [14], [22], [24] require $10^3$–$10^5$ collected packets depending on the number of bits used for marking and awareness of network topology.

An important assumption in PPM is that packets in the flow of interest are much more frequent than other normal packets [9]. Otherwise, PPM will incur a long completion delay or even fail for the path reconstruction under low-frequency traffic scenarios. Another shortcoming of PPM is that it is difficult to identify the origin of a single packet. While for applications such as attack mitigation, fault diagnosis [3] or path validation [5], it is preferable to achieve fast traceback as well as single packet traceback [17]. The above challenges in marking-based approaches motivate us to design a novel robust traceback acceleration mechanism with the ability to trace a single packet as presented in this work.

### B. Logging-based Approaches

Logging-based traceback [25] involves the storing of packet digests at intermediate routers on the path toward end-hosts,
thus achieving single packet traceback. Zhang et al. [26] presented a topology-aware single packet IP traceback system. The main disadvantage of logging-based traceback lies in that large storage space is required for packet logs [27]. To reduce the storage requirement for logging, Lee et al. [28] proposed flow digesting on routers instead of logging individual packets. SampleTrace [29] is another flow-level logging method using existing xFlow (sFlow, NetFlow and IPFIX) function and BGP information to implement traceback.

C. Hybrid Approaches

Hybrid approaches [8], [17], [30] take advantages of both packet marking and logging, to reduce the number of marked packets when conducting the traceback process and alleviate the high storage overhead at routers. Duwairi et al. [8] proposed two hybrid traceback schemes, distributed link-list traceback (DLLT) and probabilistic pipelined packet marking (PPPM), to reduce the number of packets needed for constructing attack paths in PPM through utilizing packet logging. In DLLT, if a router decides to mark a packet, it first stores the marking information which was written by the previous marking router, and then marks the packet by overwriting the marking field with its IP address. A link list is therefore established to guide the marking information collection from the end-host. PPPM is a logging-assisted marking scheme, which loads traceback messages into packets going to the same destination of these traceback messages. Gong et al. [17] presented a hybrid solution, called HIT, which reduces the storage overhead at routers to one half and could track a single IP packet. The basic idea of HIT [17] is to recursively mark the accumulated information of multiple routers on packets, and log these accumulated path information at some of the routers on the path. Nevertheless, HIT requires relatively large marking field per packet and high storage on logged routers, since the logging is performed on a per-packet basis. RIHT [30] is a hybrid IP traceback scheme for efficient packet logging aiming to have a fixed storage requirement.

The proposed traceback message delivery scheme in PPPM [8] is a related work to our design. However, several fundamental differences exist between PPPM and our design. First, PPPM assumes that IP header has enough space to hold the traceback message of a router (e.g., it requires 57 bits marking space in PPPM). While OPM/AOPM is designed based on a general message model, and message fragmentation is explicitly supported. Second, we consider the trigger-based traceback, and distinguish the internal-flow and external-flow, which is different from PPPM. Same as most existing PPM methods, PPPM traces all traffic flows. Thus, it marks every passing packet in a probabilistic manner. In OPM/AOPM, once routers are triggered to generate traceback messages, they mark passing packets in a deterministic manner until all message fragments are delivered to the end-host. Importantly, their message delivery schemes are different. In PPPM, a router always swaps the marking information of a received marked packet with one of its buffered traceback messages. The frequent swapping operation incurs very high router processing overheads and complicates the implementation. In Section VIII, we show their detailed performance comparisons.

III. PRELIMINARIES

We consider the trigger-based on-demand traceback. The triggering of a traceback procedure needs to meet the following two conditions: 1) The traceback mechanism is initiated when there appears one or more suspicious flows [9]–[11], [31] or when there is a need for network analysis [3], [4] (i.e., routers are configured to initiate traceback for matching traffic flows of interest); and 2) Routers witnessing the flows of interest decide to perform traceback (e.g., either probabilistically or periodically). Recent advances and developments in software-defined networking (SDN) make the trigger-based traceback easily implementable on SDN-enabled routers [32]–[35].

For the sake of brevity, we use the term “a flow of interest” to represent a packet or flow to be traced by a traceback mechanism. We define packets constituting the flow of interest as the internal-flow, and the other packets in the network as external-flows. In this work, we consider the general-purpose traceback for tracing the complete network path followed by a flow of interest, where the solution is also applicable for locating the source or partial path of a flow. The flow of interest could be specified by a source-destination pair or a five-tuple flow ID (srcIP, dstIP, srcPort, dstPort, protocol) [11]. With respect to a router $R$, we use the term upstream routers of $R$ to refer routers located on the path between the source of a flow of interest and $R$; the downstream routers are routers between $R$ and the end-host.

A. Traceback Message

In this work, we do not specify the length of available space for traceback marking in IP header. Without loss of generality, we assume there are $K$ bits in an IP packet header that can be typically used to encode information by a router. For example, the authors in [9], [18], [24] identified a total number of 25 bits in IP header that can be reused for marking, including the rarely used 16 bits ID field, 1 bit fragmentation flag, and 8 bits type of service. In IPv6 networks, two fields of IPv6 header, i.e., Flow Label (24 bits) and Hop-by-Hop options (8 bits) can be potentially used for traceback marking [36]. Furthermore, additional header in IPv6 can be created for marking purpose [37].

![Fig. 1. A traceback message is partitioned into a sequence of fragments](image)

Let $M_r$ denote the traceback message that router $R$ will send to the end-host $D$. Since the length of $M_r$ is normally larger than $K$ [18], it needs to be partitioned into a sequence $N$ of non-overlapping message fragments $\{m_1, m_2, \ldots, m_N\}$ so that one fragment can be embedded into one IP packet header, as shown in Fig. 1. In order to reconstruct the message $M_r$ at the end-host, we need an associative address for these fragments (i.e., message identifier) and a fragment offset indicating...
the index of a fragment. In addition, to group all relevant traceback messages along a routing path regarding a flow of interest, we also need another associative address for these messages (called capture identifier in this work). Although certain overhead is introduced by message fragmentation, message identifiers serve both as associative addresses and partial data integrity verifiers. They can potentially reduce the ability of adversaries to interject false fragments that collide with legitimate ones [18]. In the event that there is enough marking space to load a complete traceback message, message fragmentation is not needed.

B. Overview of Marking-based Traceback

Assume a traceback-enabled router $R$ captures a packet of interest $P$, and thus decides to send its message $M_r$ to the destination of $P$. Router $R$ first generates a random number as the associative identifier of $M_r$, then breaks $M_r$ into $N$ non-overlapping message fragments $\{m_1, m_2, \ldots, m_N\}$, and stores them in its local buffer [18]. These traceback message fragments will be delivered to the destination of $P$ through packet marking. For example, when router $R$ receives a subsequent packet in the flow of interest, it loads one message fragment into the packet’s IP header. This process is repeated until all message fragments have been transmitted to their destinations. After these traceback message fragments have been collected at the end-host, the network path of packet $P$ could be easily identified based on the capture identifier and message identifier embedded in message fragments.

C. Layered Architecture for Marking-based Traceback

We decouple the traceback message content encoding and delivery functions in MBT, and use a two-layered architecture to model the procedure of MBT, as illustrated in Fig. 2. The traceback decision making module decides whether a router sends its traceback message to an end-host or not. The message content encoding layer determines the content of a traceback message, including the representation and encryption of router identity information. For example, a router can choose to convey the hash value of its IP address [14] or an internal ID that is unrelated to the IP address [11], [12] to end-hosts. Encryption is to prevent tampering of markings [3], [14], [27]. One important design goal in the upper layer is to reduce the traceback message length while providing certain desirable properties such as authentication or privacy-preserving. The transmission layer in Fig. 2 takes charge of the delivery of traceback messages to end-hosts. Interesting research problems in this layer include minimizing the number of packets required to convey traceback messages [3] in PPM, achieving fast delivery of traceback messages to their intended destinations, or efficient coding technique to combat packet losses during the message delivery, etc.

As the space required for marking a traceback message is generally larger than the number of marking bits available in IP header, fragmentation is then needed so that a small enough fragment can be loaded into one IP packet header. In this work, we focus our investigation on the transmission layer, where the objective is to achieve fast and robust delivery of message fragments. Since the message content encoding and transmission layers have independent objectives, the design for the fragment delivery can be generally applied to existing MBT schemes.

IV. OPPORTUNISTIC PIGGYBACK MARKING: THE BASIC DESIGN

For the purpose of illustrating the proposed opportunistic piggyback marking, in this section we first present the basic design known as OPM. In Section VI, we extend this basic design to fully utilize free ride opportunities for robust delivery of traceback messages.

A. Motivation

The limitation of traceback message delivery in existing MBT methods stems from the implicit assumption that message fragments are only carried by the packets that belong to the flow being traced [18]. As described in Section III, since a sequence number of packets is needed to convey a single traceback message to the destination, it may take a long time for an end-host to collect all traceback messages from routers to reconstruct the network path of interest. In this section, we discuss the situation that fragments buffered in individual routers can be delivered faster to the end-host without incurring extra message overhead. We term such situations as the piggyback marking opportunities.

Fig. 3 shows an example of direct piggyback marking. In this example, the flow of interest is forwarded from $S$ to $D$ through intermediate routers $\{R_1, R_2, R_3, R_4, \ldots\}$. Besides the traffic from $S$ to $D$, $R_5$ and $R_6$ are forwarding packets to the destination $D$. Assume all routers along the network path are traceback-enabled routers, and are thus involved in the packet marking operation. From Fig. 3, we observe that packets in external-flows forwarded
from \( R_5 \) or \( R_6 \) can carry these message fragments directly to \( D \). The delivery delay of traceback messages can be opportunistically reduced by exploiting external traffic flows going to \( D \). We call it as the direct piggyback marking.

### B. Design Description

1) **Marking Field Allocation in OPM:** We assume that the IP header provides \( K \) bits for the purpose of traceback marking. The marking field allocation in OPM is listed in Table I.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI</td>
<td>( c )-bit capture identifier to link traceback messages from different routers for a given flow of interest</td>
</tr>
<tr>
<td>MI</td>
<td>( l )-bit message identifier to link message fragments for a specific router</td>
</tr>
<tr>
<td>FO</td>
<td>( \lceil \log_2 N \rceil )-bit fragment offset, where ( N ) is the number of fragments for a message</td>
</tr>
<tr>
<td>FP</td>
<td>( (K-c-l-\lceil \log_2 n \rceil))-bit for the fragment payload, where one fragment of a traceback message is loaded</td>
</tr>
</tbody>
</table>

To ensure correct reassembling of message fragments and identification of a network path at end-hosts, the control information CI, MI and FO are necessary in all MBT methods with message fragmentation [18]. In other words, OPM does not introduce extra control information overhead. For example, let us assume 25 bits are available for marking in IP header [24], and CI and MI are assigned 5 bits each. There will be 15 bits remaining for FO and FP. Note that the content of a traceback message is determined by the message encoding layer, and thus is independent of the marking field allocation. For the traceback message fragmentation, given \( K \)-bit available marking space, \( c \)-bit CI, \( l \)-bit MI, and the traceback message length \( L \), we can derive the minimum number of fragments \( N \), by solving (1).

\[
\min N, \\
\text{subject to } \lceil \log_2 N \rceil + \left\lceil \frac{L}{N} \right\rceil \leq (K-c-l) \tag{1}
\]

2) **Traceback Message Triggering:** Our traceback objective is to let the end-host reconstruct the network path that a specific flow of interest traverses. Therefore, all traceback-enabled routers along the path will be involved in the traceback procedure. We illustrate the main idea of OPM through an example shown in Fig. 4. Suppose the first traceback-enabled router \( R_1 \) captures a packet \( P \) in the flow of interest and decides to perform traceback accordingly at time \( t_1 \). It generates a traceback message \( M_{t_1} \) to be transmitted to the destination of \( P \), which is divided into \( N \) non-overlapping message fragments. Router \( R_1 \) computes a random number as the capture identifier CI. Here, we can use the TTL value in \( P \) as the message identifier MI, where the relative hop distance information also indicates topological order of traceback messages.

Thus, \( R_1 \) sets MI using the TTL value of \( P \) for all relevant message fragments, so that they can be easily reassembled at the end-host. Then, router \( R_1 \) writes the first fragment into the IP header of \( P \). We define the packet that carries the first message fragment from the first traceback-enabled router as the traceback trigger signal TTS, which is to trigger downstream traceback-enabled routers to generate their respective traceback messages regarding \( P \), and buffer them for subsequent transmissions to the intended destination. Note that our traceback scheme captures snapshots of the flow of interest. Only the TTL value present in a TTS is used as the message identifier for relevant message fragments when they are initially generated at a router. Thus, even subsequent topology or routing path (e.g., due to multipath routing) changes for the flow of interest, relevant message fragments can be successfully reassembled since they have the same message identifier.

![Fig. 4. Example of the packet marking procedure in OPM](image)

Algorithm 1 (Lines 2-7) describes the traceback message triggering process in OPM. We define the function IsMarked(\( P \)), which checks whether a received packet \( P \) is a marked packet or not. If all OPM related marking fields in \( P \) have zero (or unrecognized) values, \( P \) is considered an unmarked packet. The function IsTTS(\( P \)) checks whether a received packet \( P \) is a TTS. If \( P \) is a marked packet, it extracts the CI information from \( P \) and checks whether there is a recent entry having the same CI value in the message fragment buffer. If not, it considers \( P \) a TTS.

For the example in Fig. 4, a downstream traceback-enabled router \( R_2 \) receives a TTS at time \( t_2 \). It generates a traceback message \( M_{t_2} \), which will be partitioned into \( N \) message fragments (Lines 5-6). From the TTS, \( R_2 \) extracts the CI and MI values for setting the control fields of these message fragments. In OPM, message fragments together with their respective destinations are stored temporarily in the local buffer (Line 7). Then, \( R_2 \) forwards this TTS to the next-hop router \( R_3 \). This process is repeated until the TTS arrives at the destination.

3) **Message Fragment Delivery:** After the traceback message triggering process, traceback messages are generated at routers on the routing path that \( P \) traverses. When a traceback-enabled router receives an unmarked packet, it checks whether this packet can carry any message fragment in the buffer to its intended destination by comparing their respective destinations. If yes, the router marks \( P \) with the first matched message fragment, which is described in Algorithm 1 (Lines 9-12).

Let us revisit the example in Fig. 4, assume \( R_2 \) receives an unmarked packet \( Q \) that goes to the same destination of \( P \) at time \( t_3 \). It marks \( Q \) with one fragment, removes the fragment together with the cached destination information from the local buffer, and then forwards \( Q \) to the next-hop router \( R_3 \). Note that when a message fragment in the buffer exceeds a
Algorithm 1: A traceback-enabled router $R$ processes a received packet in OPM

1  Procedure: Receive($Packet \ P$)
2     if IsMarked($P$) == TRUE then
3         if IsTTS($P$) == TRUE & & $P$.dest $\neq R$.addr then
4             /*$P$ is a TTS and $R$ is not the destination of $P$*/
5                 M=GenerateMessage($P$);
6                 F=Fragmentation($M$);
7                 Store($F$, $P$.dest);
8             else if IsCarrier($P$) == TRUE then
9                 /*Check whether $P$ can carry any fragment in the buffer to its intended destination*/
10                f=GetFragment($P$);
11                Mark($P$, f);
12                Forward($P$);
13  end;

particular time limit, it will be removed from the buffer. In case a router receives a marked packet (but not a TTS) or decides not to mark a packet, it just forwards the packet.

4) Message Reconstruction: When the end-host receives a marked packet, it will extract the message fragment from the received packet before sending it to upper layers for further processing. Given a collection of message fragments associated with a specific TTS received at the end-host, the message reconstruction is based on a combinatorial process [18]. The end-host first classifies all received traceback message fragments according to their capture identifiers. Then, it groups the message fragments with the same message identifiers (i.e., the relative hop distance value) in the right order based on the fragment offset. Finally, the end-host recovers all the traceback messages, where the topological order of routers can be derived based on the relative hop distance value.

V. NUMERICAL ANALYSIS

In this section, we present theoretical analysis of OPM. The obvious queueing behavior in OPM motivates us to develop a queueing theory based analytical model to capture its performance. We model OPM for tracing back a particular flow of interest at a traceback-enabled router using an $M^X/M/1/C$ finite queue with batch arrivals [38]. Our analytical model can be easily extended to scenarios of tracing back multiple flows of interest considering that traceback triggering of these traffic flows is independent of each other. Here, traceback message fragments are treated as clients and packets that carry these fragments collectively act as a virtual server in the queueing system.

A. $M^X/M/1/C$ Queueing Model

1) Arrival Process: When a traceback message is generated at a particular traceback-enabled router, a sequence of message fragments will be temporarily inserted into the buffer and then transmitted to the destination. Network arrivals have often been modeled as Poisson processes for analytic simplicity [39]. Since the arrival of traceback messages is triggered by the appearance of traffic flow of interest, we model the arrival of traceback messages as a Poisson process. As aforementioned, the space required by a traceback message is typically larger than the available bits for marking in each IP packet. Given a traceback message encoding scheme, a traceback message will be partitioned into fixed number of fragments. Therefore, such a process can be modeled as a fixed-size batch arrival process by treating each fragment as a unit (customer) in the queueing system.

2) Departure Process: Once a packet that can carry a fragment to its intended destination (referred to as a carrier) arrives, a customer (i.e., one message fragment) gets the service and then leaves the system. Considering packets arrive in sequential order at a router, for a particular group of fragments, we treat their carriers as a virtual server, which serves one customer at a time from the front of the queue. Different from the departure process in traditional queueing systems, the customer in MBT leaves the system immediately after the server becomes available at the queue. For easy understanding, we define the service time of a fragment as the time period from the fragment being selected as a to-be-served customer (i.e., being as the first item in the service queue) to the time a carrier arrives at the queue. In this way, the service time is captured by the inter-carrier arrival time, which is assumed to have an exponential distribution.

3) Finite Capacity System: In any practical system, the buffer size for traceback message fragments in a router is limited. Therefore, considering the arrival and departure processes discussed above, we model the OPM process at individual routers as an $M^X/M/1/C$ bulk queue with finite capacity, where the queueing discipline is first-match-first-serve.

B. Stationary Analysis

For clarity, Table II lists and briefly describes the definitions used in the model.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>average arrival rate of traceback messages</td>
</tr>
<tr>
<td>$\mu$</td>
<td>average arrival rate of carriers (i.e., service rate)</td>
</tr>
<tr>
<td>$n$</td>
<td>batch size, i.e., number of fragments per message</td>
</tr>
<tr>
<td>$\rho$</td>
<td>traffic utilization, which is defined as $\frac{\lambda}{\mu}$</td>
</tr>
<tr>
<td>$C$</td>
<td>service queue size</td>
</tr>
<tr>
<td>$k$</td>
<td>system state, which is represented by the number of traceback message fragments in the buffer steady-state probability, which is the probability that there are $k$ fragments in the system in steady state</td>
</tr>
<tr>
<td>$p_k$</td>
<td>system size, mean number of fragments in the system</td>
</tr>
<tr>
<td>$W$</td>
<td>mean waiting time of a fragment in the system</td>
</tr>
<tr>
<td>$D$</td>
<td>drop probability of the system</td>
</tr>
</tbody>
</table>

Fig. 5. A Markov model for the marking-based traceback
whole. The state transition diagram for a single-server queue with finite capacity and batch arrivals is shown in Fig. 5. In equilibrium ($\rho < 1$), we have the following balance equations for Markov chain model [40],

$$\begin{align*}
\lambda p_0 &= \mu_1 \quad k = 0, \\
\mu_{p_{k+1}} &= (\lambda + \mu)p_k \quad 1 \leq k < n, \\
\lambda p_{k-n} + \mu_{p_{k+1}} &= (\lambda + \mu)p_k \quad n \leq k \leq C - n, \\
\lambda p_{k-n} + \mu_{p_k} &= \mu_k \quad C - n < k < C, \\
\lambda p_{k-n} &= \mu_k \quad k = C,
\end{align*}$$

where $\sum_{k=0}^{C} p_k = 1$. We proceed to define the performance metrics in our model.

2) Steady-state System Size: One of the important metrics to evaluate the OPM performance is the steady-state system size, which measures the number of fragments in the system. By solving the set of balance equations given in Eqs. (2), the steady-state probabilities can be obtained. Then, we have the steady-state system size,

$$\mathcal{L} = \sum_{k=0}^{C} k \cdot p_k$$

3) Average Waiting Time: According to Little’s Law in queueing theory [38], $\mathcal{L} = \frac{\lambda n \mathcal{W}}{\lambda n}$. Then, we can derive the average waiting time of a fragment in the system from the time of its insertion into the queue to the time it is served,

$$\mathcal{W} = \frac{\mathcal{L}}{\lambda n}$$

4) Drop Probability: For a finite queue, drop rate of the system is an important metric. We have the probability that an arriving batch of message fragments is dropped,

$$\mathcal{D} = \sum_{k=C-n+1}^{C} p_k$$

C. Numerical Results

We analyse the buffer system of OPM at a router. Numerical results are calculated by the mathematical software Sage [41], and are further verified by a customized time-driven simulator for the $M^X/\text{M}/1/C$ queue model.

1) Impact of Piggyback Marking Opportunity: In this experiment, we set the maximum queue length $C$ to 100 fragments, and the batch size $n$ to 5. The arrival rate $\lambda$ is set to 0.1. Let us assume the internal-flow arrival rate is 0.6. Here, the internal-flow refers to packets constituting the flow of interest, and the other packets are referred to as external-flows. We vary the arrival rate of external-flow that has the same destination of the buffered message fragments from 0 to 0.4 (denoted as $\Delta \mu$). That is, the service rate $\mu$ increases from 0.6 to 1.0. This setting examines the impact of the increase of piggyback marking opportunities on the performance of the buffer system. Note that when $\Delta \mu$ is set to 0, it is the basic message delivery scheme where only internal-flow can be exploited as carriers to deliver the message fragments.

Fig. 6(a) and Fig. 6(b) plot the average steady-state system size and waiting time against piggyback marking opportunity, respectively. As expected, the average system size decreases from 14 to 3 with the increase of $\Delta \mu$. Correspondingly, the average waiting time of a fragment in the buffer drops from 17.6 to 6.0 time units. The simulation results, which have been averaged over 1000 runs and the corresponding standard deviations are provided as error bars, match our analysis closely and thus verify the analytical results. In addition, we observe from the simulation results that, as $\Delta \mu$ increases, the variations of queueing performance significantly decrease. Since the maximum queue length is much larger than the batch size and the traffic utilization $\rho$ is always smaller than one, the drop probability in this test is close to zero.

2) Impact of Maximum Queue Length and Batch Size: Fig. 7 depicts the numerical results of average waiting time and drop probability when varying the maximum queue length and batch size, where $\lambda$ and $\mu$ are fixed to 0.1 and 1.0, respectively.

From Fig. 7(a), when the batch size is 5 with different maximum queue lengths, the average waiting time does not change much, and the drop probability is negligible as shown in Fig. 7(b). However, when the batch size is raised to 9, the average waiting time increases significantly from 15.7 to 34.1 units of time with increasing the maximum queue length. At the same time, when the maximum queue length is 40, the drop probability is increased up to 8%. Even as the maximum queue length is increased to 100, the drop probability is still non-negligible around 2%. We observe that given a queue with limited capacity, large batch size may lead to long waiting time and high drop probability.

3) Summary: The above results indicate that, in OPM, the average number of fragments and waiting time of a fragment in the buffer can be effectively reduced with high traffic rate of the external-flow, and thus achieving a fast IP traceback. In addition, from the simulation results shown in Fig. 6, exploiting piggyback marking opportunities can decrease the performance variance of MBT. Therefore, OPM potentially provides more stable performance guarantee. However, we
also observe from Fig. 7 that, OPM still suffers performance degradation in case of high traffic intensity with a large $\rho$ value (e.g., the service rate is low) or limited capacity of the buffer. We will address this issue in the next section.

VI. OPPORTUNISTIC PIGGYBACK MARKING: THE ADVANCED DESIGN

When the piggyback marking opportunities are insufficient, there is a possibility of message fragments waiting for a long time in the buffer or being dropped due to overflow. To overcome this issue, in this section, we present the advanced design of opportunistic piggyback marking (referred to as AOPM) for IP traceback.

A. Motivation

Before providing the detailed design, we first illustrate the motivation behind AOPM. In Fig. 8, we demonstrate how fragment delivery can benefit from what we call the one-hop piggyback marking. Suppose there are some message fragments buffered at $R_2$ to be sent to $D$. However, traffic rate of the flow going to the destination is low within a certain time frame. In this case, we exploit the one-hop piggyback opportunity to move forward each message fragment hop by hop. As shown in Fig. 8, the packet sent from $R_6$ to $R_7$ can opportunistically carry a message fragment from $R_2$ to $R_3$. Since path segments closer to the destination are expected to be shared by more paths to the destination, one-hop piggyback marking potentially increases the probability that message fragments encounter direct piggyback marking opportunities in downstream path segments.

![Fig. 8. Example of one-hop piggyback marking](image)

While the mechanism of one-hop piggyback marking is intuitive, simply applying a greedy strategy and exploiting every one-hop piggyback marking opportunity incurs larger processing overhead on routers, and may miss the subsequent direct piggyback marking opportunities. To efficiently utilize potential piggyback marking opportunities, we propose a delay-driven opportunistic piggyback marking for IP traceback, where piggyback marking is driven by a designated delay bound.

B. Design Description

1) Two Piggyback Marking Modes: We define two piggyback marking modes: direct piggyback mode and one-hop piggyback mode. By default, AOPM works in the direct piggyback mode. Routers load the traceback message fragments into unmarked passing packets that can directly carry these fragments to their intended destinations. If a fragment staying in the local buffer has exceeded a specified delay bound, e.g., there is very sporadic traffic going to the destination of this fragment, the fragment enters into the one-hop piggyback mode. In this case, this fragment will be carried forward to the next-hop router along the network path of interest. Then, it returns to the direct piggyback mode and waits for direct piggyback marking opportunity. There exists a tradeoff between traceback delay and processing overhead in these two modes. In AOPM, we use an adjustable parameter (i.e., the delay bound) to control the tradeoff between delay and processing overhead.

2) Traceback Triggering: The marking field allocation in AOPM is same as that in OPM. Similarly, TTS is used to trigger downstream traceback-enabled routers to send their traceback messages to the target. To achieve efficient piggyback marking in AOPM, we specially introduce the metadata cache and two message fragment queues (i.e., direct piggyback queue and one-hop piggyback queue) in each traceback-enabled router. In Fig. 8, assume the first traceback-enabled router $R_1$ has a traceback message $M_{r_2}$ for the packet of interest $P$ to be transmitted. Before router $R_1$ sends out the TTS, it stores a five-tuple in the metadata cache, which includes CI, MI, the next-hop and destination addresses of packet $P$, and the timestamp. Due to the limited cache size, non-updated entries in metadata cache will be removed once they exceed a time limit. Since the first message fragment from $R_1$ is piggybacked on TTS, the direct piggyback queue at $R_1$ initially holds the remaining message fragments temporarily until they can be processed later.

The procedure of processing a received packet in AOPM is described in Algorithm 2. For the example in Fig. 8, once a downstream traceback-enabled router $R_2$ receives a TTS, it updates metadata cache with the five-tuple information extracted from the TTS. Then, router $R_2$ generates a traceback message $M_{r_2}$, and all message fragments of $M_{r_2}$ will be stored in the direct piggyback queue (Lines 4-8). After that, router $R_2$ will forward this TTS to the next-hop router as usual.

In terms of the queue management for message fragments, when a buffered fragment has been in the queue beyond the mode switching expiration period $T_{ms}$, it will be removed from the direct piggyback queue, and inserted into the one-hop piggyback queue. For both direct piggyback queue and one-hop piggyback queue, message fragments that exceed a designated delay bound $T_d$ will be dropped from the queues. $T_{ms}$ and $T_d$ are two adjustable system parameters in AOPM.

3) Message Fragment Delivery: Algorithm 2 (Lines 9-20) describes how a traceback-enabled router $R$ processes a received packet $P$ after the traceback event has been triggered. If $P$ is a marked packet but not considered a TTS, $R$ first checks whether the message fragment marked in $P$ is just for one-hop delivery (Line 9). In function IsMarkedForOnehop($P$), to derive the piggyback marking mode of the message fragment marked in $P$, $R$ first extracts the capture identifier CI from the AOPM marking fields in $P$ and checks whether there is a hit in the metadata cache. In the event that there is a hit, if the marked packet and message fragment have different destinations, that means the packet is being used as a one-hop carrier. In this case, the function returns TRUE. Then, router $R$ extracts the marking information, generates a fragment and...
Algorithm 2: A traceback-enabled router $R$ processes a received packet in AOPM

1. **Procedure: Receive**($Packet P$)
2. **if** $IsMarked(P) == TRUE & & P.dest \neq R.addr$ **then**
3. // $P$ is a marked packet and $R$ is not the destination of $P$
4. **if** $IsTTS(P) == TRUE$ **then**
5. UpdateMetadataCache($P$);
6. $M = GenerateMessage(P)$;
7. $F = Fragmentation(M)$;
8. DirectQueueInsert($F$);
9. **else** **if** $IsMarkedForOnehop(P) == TRUE$ **then**
10. $f = ExtractMark(P)$;
11. DirectQueueInsert($f$);
12. ResetMarkingFields($P$);
13. **if** $IsMarked(P) == FALSE$ **then**
14. **if** $IsOnehopQueueCarrier(P) == TRUE$ **then**
15. $F = GetFragmentFromOnehopQueue(P)$;
16. **else** **if** $IsDirectQueueCarrier(P) == TRUE$ **then**
17. $F = GetFragmentFromDirectQueue(P)$;
18. **if** $f == NULL$ **then**
19. $Mark(R, f)$;
20. Forward($P$);

inserts it into the direct piggyback queue. The newly inserted fragment returns to the direct piggyback mode and waits for direct piggyback marking opportunity (Lines 9-11). After that, $P$ is set as an unmarked packet (i.e., reset AOPM marking fields to zeros), and it goes into the following process as if $R$ receives an unmarked packet (Line 12). Otherwise, if there is no hit, the router will forward this packet to the next-hop.

When $R$ receives an unmarked packet $P$, it first checks whether $P$ can carry any message fragment in the one-hop piggyback queue to its intended next-hop router or the destination in a first-match first-serve manner (Line 14). This can be achieved by comparing the next-hop of $P$ and the next-hop of the message fragment in the metadata cache. If there is no message fragment in the one-hop piggyback queue that $P$ can carry, the router will check whether any message fragment in the direct piggyback queue goes to the same destination as $P$ (Line 16). Upon finding a message fragment in either queues, the router will mark $P$ with the message fragment, and remove the selected message fragment from the corresponding queue. Then, $R$ forwards $P$ to the next-hop router.

4) Message Reconstruction: At the end-host, the message reconstruction process is same as that in OPM. The details are given in Section IV.

C. Design Discussions

1) Traceback Decision Making: Although our design is independent to the traceback decision making, we want to point out that for tracing the sources of high volume traffic flows, tracing every single packet (i.e., sending a TTS for every packet in OPM/AOPM) is neither practical nor necessary. It is more efficient that traceback decision is made at an individual router either probabilistically or periodically. For example, under a DDoS attack, strategic routers installed with intrusion detection systems may detect anomalies [42], and thus initiate the traceback based on a probability measure. Similarly, intermediate routers can probabilistically generate traceback messages after receiving a TTS to reduce overhead. In the event that multiple routers trigger traceback for the same flow within a short time interval, downstream routers will receive the TTSes sent from upstream routers and learn that the traceback has been initiated by one or more upstream routers. At a subsequent period, downstream routers will be suppressed to generate TTSes to avoid duplicated traceback decisions.

2) Single-packet Traceability: In our design, once a traceback decision is made, a TTS is used to trigger all traceback-enabled routers on the way to generate traceback messages to its destination in a deterministic manner. Therefore, OPM/AOPM is able to trace a single IP packet. Such single-packet traceability is essential for both tracing DoS attacks which are composed of only a few numbers of packets (e.g., software exploit attacks in [19]) and diagnosing network issues [17], [26].

3) Queuing Model of AOPM: We introduce two queues at a single node for AOPM, one for direct piggyback, and another for one-hop piggyback. To model AOPM, we can combine these two queues to form a priority queue with two kinds of customers, where each message fragment joining the queue has an associated deadline. We call message fragments in direct piggyback mode and one-hop piggyback mode as class 1 and class 2 customers, respectively. In such a queue, class 2 has non-preemptive priority over class 1, and the arrival of class 2 customers is dependent of the mode switching expiration of message fragments in the queue. Priority queues [43], queues with job deadlines [44] as well as queues with customer transfers [45] have been studied extensively, but separately. The hybrid system of priority queue with customer transfer and job deadlines presents a complex structure. To the best of our knowledge, there is no known (product form) solution for the stationary analysis of such hybrid system [46]. There are also two kinds of servers depending on whether they go to the intended destination or not. Having two kinds of servers makes the model even more difficult to analyze. We therefore evaluate AOPM using extensive simulations (in Section VIII), leaving the queuing analysis of AOPM for future work.

4) Identifier Collision Resolution: Due to limited marking space in IP header, the length of capture identifier (CI) for associating traceback messages is normally short (e.g., 8 bits or less). However, identifier collision risk at an end-host increases with decreasing the length of capture identifier. One possible solution is to exploit packet inherent features (such as checksum pattern) as the second-factor associative identifier. When delivering a group of traceback messages along a routing path, routers only mark those packets that exhibit similar features. Then, at the end-host, combinatorial search with considering packet features (i.e., clustering the received message fragments) is used to reconstruct traceback messages, and thus reducing the collision possibility. Resolving identifier collision in MBT will be an interesting topic, but detailed study is beyond the scope of this paper.
VII. FLEXIBLE MARKING-BASED TRACEBACK FRAMEWORK

In this section, we present the Flexible Marking-Based Traceback (FMBT) framework, a novel design paradigm for marking-based IP traceback.

A. Framework Overview

The FMBT framework is illustrated in Fig. 9. In contrast to the prior art [7], [8], [10], [14], [16], [21], [47], traceback messages in FMBT can be delivered to any designated end-host. As shown in Fig. 9, routers that witness the suspicious flow will send traceback messages to a traceback server in the Cloud. For example, ISP can deploy such traceback servers to offer query-based traceback services to end-users. When the end-host (e.g., victim) needs to retrieve any relevant traceback information, it will launch traceback queries to the server. Our OPM/AOPM design offers flexibility to deliver traceback messages to any designated address in MBT, which facilitates the implementation of an efficient FMBT system without incurring extra message overhead.

Fig. 9. Flexible marking-based traceback framework

As mentioned earlier, an intrusion detection system or network operator may trigger the traceback mechanism. Along with the event triggering, the address of the traceback server is specified. Consider Fig. 9 for example. The first traceback-enabled router \( R_1 \) marks a packet of interest and specifies this packet as the traceback message signal (TTS). This TTS carries the capture identifier and the address of the designated destination, and finally arrives at the end-host. Note that the TTS does not carry any message fragment in FMBT, since all message fragments will be sent to the designated server, rather than the victim. As TTS is forwarded in the network, downstream traceback-enabled routers along the attack path are notified to send their traceback messages to the designated server. Once a downstream traceback-enabled router \( R_2 \) receives a TTS, it generates a traceback message, and performs the message fragmentation for later delivery. When the end-host receives a TTS, capture identifier together with the timing information can be used for accessing traceback information in the server. Meanwhile, routers that have buffered traceback message fragments explore direct or one-hop piggyback marking opportunities to transmit these fragments to the server, which is same as that in OPM/AOPM.

B. Features of FMBT Framework

The FMBT framework has several favourable features that previous single traceback scheme failed to satisfy simultaneously. We list the potential benefits offered by FMBT framework as follows.

1) Privacy Preserving Traceback: One main challenge hindering the practical deployment of traceback is the security concern of disclosing the network topologies from ISPs. Unfortunately, traditional MBT approaches send marking information to end-hosts, and thus have the risk of conveying commercially sensitive router information outside ISP. For example, an adversary may misuse the traceback mechanism to infer ISP’s network topologies from the received packets with marking values [27]. On the contrary, FMBT has intrinsic privacy-preserving property as it can store traceback information in trusted traceback servers. These traceback servers will be managed by ISPs themselves, and thus sensitive information could be adequately maintained by each individual ISP.

2) Traceback for Forensic Analysis: In FMBT, the traceback marking information could be stored longer in traceback servers compared with the traditional MBT approach. Therefore, FMBT enables forensic analysis which can be performed postmortem, i.e., investigation of network traffic a posteriori to collect sufficient evidence that allows the perpetrator to be prosecuted.

3) Incentive for Traceback Deployment: Since IP traceback techniques require active participation of routers, ISPs are positioned to potentially play a vital role in a traceback system. One reason impeding traceback being broadly deployed on the Internet is that ISPs lack economic incentives to deploy traceback in their networks.

FMBT allows deploying traceback as a charged service, which encourages ISPs’ involvement to deploy the traceback service in their networks. For instance, ISPs may charge a fee to customers (e.g., networks, end-users, or law enforcement agencies) who are interested in obtaining traceback information. Given the capture identifier which is conveyed by TTS to the end-host, and an approximate time of the traffic, paid customers can retrieve the relevant traceback information by sending queries to the server.

4) Scalable and Robust Traceback: In existing MBT methods, all traceback messages are transmitted to the destination of traffic flow being traced (e.g., the victim). It may work well when IP traceback is used for non-adversarial network analysis applications with low traffic intensity. However, to trace back adversarial attacks, it inevitably incurs a heavy burden on the victim by requiring it to log all the traceback message fragments and then to reconstruct the network path. The situation becomes worse if the victim’s limited resources have been saturated by the attack traffic [42]. In contrast, FMBT shifts the overwhelming storage and computational burden from the victim to resource-rich servers, thus increasing the scalability and robustness of marking-based IP traceback.

VIII. EVALUATION

In Section V, we have shown the theoretical analysis of OPM at individual routers, and quantitatively demonstrated
the benefits of opportunistic piggyback marking to MBT, such as achieving fast IP traceback, reducing fragment buffering time at router’s local memory and decreasing drop probability. In this section, we evaluate OPM and AOPM for network scenarios using the ns-2 [48] simulator. The main objective is to show how fast and robust traceback messages can be delivered to the end-host. Since the message delivery is orthogonal to the traceback message content, we examine the traceback message delivery performance without specifying message content encoding schemes.

A. Simulation Setup

In each simulation experiment, we trace the routing path of a flow by specifying the source-destination pair in a linear network. Links between routers are set up as duplex with 1ms delay and 100 Mbps network capacity. Exponential traffic generator is used to generate on/off traffic, where burst times and idle times are taken from exponential distributions. In order to reflect realistic traffic profiles, traffic rates in simulation are generated by randomly sampling the source-destination data rates from CAIDA DDoS Attack 2007 Dataset [49]. The average data rate per source-destination pair sampled from the dataset is around 40 kbit/s, where DDoS attacks usually consist of a large number of small flows [50]. We use the default Static Routing protocol and DropTail queue policy in the ns-2 simulator.

Background traffic (i.e., external-flows) is generated along the flow of interest. To simulate different traffic distribution scenarios, we generate two kinds of background traffic flows: 1) random-source but fixed-destination flows, and 2) random source-destination pair flows. Unless otherwise specified, the fragment buffering expiration bound \( T_d \) is set to 2.0s. The packet size is set to 250 bytes and the queue length at each router is set to 1000 packets. The flow of interest lasts 20s and background traffic flows last 50s. We vary the number of fragments per traceback message from 5 to 20, and the path length from 10 to 25 hops. The evaluation results have been averaged over 500 runs.

We compare the performance of following message fragments delivery schemes, and the same traceback triggering mechanism is applied in these schemes.

- **OPM**: the basic design of opportunistic piggyback marking, where all packets with the same destination (regardless of source address) as the flow being traced can be used to deliver message fragments.
- **AOPM**: the advanced design of opportunistic piggyback marking with the one-hop piggyback mode enabled.
- **SWAP**: the traceback message delivery scheme in PPPM [8]. When a router \( R \) receives an unmarked packet \( P \), if there is a local message fragment to be transferred to \( P \)'s destination, \( R \) marks \( P \) with the message fragment. When \( R \) receives a marked packet \( P \), if there is a buffered message fragment to be transferred to \( P \)'s destination, \( R \) always swaps \( P \)'s marking information with the message fragment in its local buffer.
- **IFM** (Internal-Flow Marking): only flows being traced can be exploited as carriers to deliver the message fragments. IFM generalizes the basic message delivery scheme (e.g., [18]) in MBT.

B. Performance Metrics

We use the following metrics for comparison.

- **Fragment Delivery Ratio**: the ratio of the number of message fragments received by the end-host to the total number of message fragments sent by routers.
- **Normalized Traceback Delay**: the time elapsed from the point when a TTS is generated by the source router to the point when the end-host receives the last traceback message fragment associated with this TTS, then divided by the fragment delivery ratio.
- **Normalized Enqueue Operation Overhead**: total number of enqueue operations incurred for transmitting message fragments triggered by a TTS divided by the corresponding fragment delivery ratio.
- **Number of Carriers**: total number of carriers that are used for delivering message fragments regarding each TTS. This metric is directly related to the performance of fragment delivery ratio and traceback delay.

C. Simulation Results: Scenario I

We first evaluate different message fragments delivery schemes under the traffic scenarios where there are sufficient direct or one-hop piggyback marking opportunities, but the traffic rate of the flow of interest is relatively low. In this experiment, the piggyback marking mode switching expiration period \( T_{ms} \) is set to 1s and each traceback message is partitioned into 10 fragments. The source-destination path length of the flow of interest is set to 25. For the background traffic, 10 flows are generated by randomly selecting source routers which are on the network path that the flow of interest traverses, but the destination is set as the end-host of the flow of interest. These flows mainly serve as the opportunistic direct carriers. The other 10 random traffic connections between any two routers are established, which opportunistically serve as the one-hop free rides.

Fig. 10 shows the performance comparison results of Scenario I where the data rate of the flow of interest is set to 50 kbit/s. From Fig. 10(a), AOPM achieves as high as 99% fragment delivery ratio. IFM show a very low 21% delivery ratio since only packets that have the same source-destination pair as that of the flow being traced can be exploited as carriers. An interesting result is that, SWAP shows 97.3% delivery ratio, which is 10% higher than that of OPM, although they both exploit the direct piggyback marking opportunities. This is because the swapping operation can move message fragments closer to the end-host, and thus increasing the marking opportunities. We can observe this phenomenon in Fig. 10(d). SWAP encounters more direct carriers than OPM and AOPM. However, apart from the direct carriers, AOPM also exploits the one-hop free rides, and it harvests 40% more piggyback marking opportunities than that of SWAP and OPM.

Fig. 10(b) plots the CDF curves of the normalized traceback delay for different schemes. The result of IFM is not shown due to its very low delivery ratio. From Fig. 10(b), SWAP
increases the traceback delay by more than 11% compared with AOPM, due to the frequent swapping operations during the traceback process. AOPM performs slightly better than OPM on average. This is because, data rate of the traced flow is relatively low, resulting a 86% delivery ratio due to insufficient direct carriers. Thus, the normalized traceback delay of OPM scheme is higher than that of AOPM.

Fig. 10(c) depicts the performance comparison on the normalized enqueue operation overhead. SWAP incurs several times higher overhead than that of OPM and AOPM, which shows SWAP to be a prohibitively expensive traceback message delivery scheme. Compared with OPM, AOPM introduces a certain extra enqueue operation overhead. However, it effectively increases the fragment delivery ratio by 15%.

To understand the difference between OPM/AOPM and SWAP during the message delivery process, we record the delivery sequence of each fragment in a simulation instance. The results are shown in Fig. 11. Router IDs from the source to destination on the path that the flow of interest traverses are indexed from 1 to 25. Due to the swapping operation, traceback messages from routers closer to the attack source always arrive at the end-host later. That means it takes longer time to locate the source during the traceback process in SWAP. From Fig. 11, we can see that message fragments from router 1 arrive last, and thus the end-host could only identify the source router ID after 1.9s. While for AOPM in this simulation instance, the last message fragment from route 1 was delivered at the end-host at 0.2s.

Next, we increase traffic rates of the flow of interest by setting the data rate of the flow of interest to 100 kbit/s. The performance comparison results are shown in Table III. We observe that, OPM, AOPM and SWAP achieve high delivery ratio, since there are enough direct piggyback marking opportunities coming from the flow of interest and background traffic flows. The only difference is that SWAP incurs 2721 number of enqueue operations, much higher than the other schemes. IFM shows lower delivery ratio and longer traceback delay due to the lack of enough packets in the flow of interest. The results also indicate that if we continue to increase the traffic rate, IFM’s performance will eventually get close to that of OPM and AOPM.

D. Simulation Results: Scenario II

To highlight the effectiveness and robustness of AOPM in random traffic scenarios, in this experiment, we remove the random-source fixed-destination flows from the background traffic. 20 random traffic connections are established between any two routers on the network path that the flow of interest traverses. The data rate of the flow of interest is set to 50 kbit/s. This traffic distribution setting corresponds to the scenario where there are sufficient one-hop piggyback marking opportunities but less direct piggyback marking opportunities.

1) Impact of Path Length: We first evaluate the impact of path length on the performance of different schemes, by varying the source-destination hop distance of the packet/flow of interest from 10 to 25. The path length is set based on the fact that most of the packets on the Internet take at most 25 hops to reach the destination [13], and the average path length is roughly 15. The piggyback marking mode switching expiration period $T_{m,s}$ is set to 1s and each traceback message is partitioned into 10 fragments by default.

In Fig. 12(a), we compare the fragment delivery ratio performance for different schemes when increasing the path length. As the path length increases, the delivery ratio of IFM drops from 56% to 21%, and OPM’s performance shows a similar decrease from 60% to 28%. However, AOPM and SWAP maintain above 98% delivery ratio regardless the change of the path length. Insufficient direct carriers lead to the poor performance of IFM and OPM, i.e., fragments will be dropped before subsequent direct marking opportunities appear.

Fig. 12(b) illustrates the normalized traceback delay as the path length varies from 10 to 25. When the path length increases, traceback delays for all schemes are increased.
However, AOPM always provides the shortest traceback delay by exploiting the one-hop piggyback marking opportunities. While AOPM and SWAP show comparable performance in delivery ratio as shown in Fig. 12(a), it is surprising to observe that SWAP yields the highest normalized traceback delay in Fig. 12(b). This is because SWAP can always prolong the survival time of message fragments. As a result, with insufficient direct carriers, fragments move slowly towards the end-host. Overall, AOPM provides obvious traceback delay reduction by 34% on average compared with SWAP.

Due to the low delivery ratios of IFM and OPM, Fig. 12(c) only reports the normalized enqueue operation overhead results for AOPM and SWAP. The number of one-hop carriers in Fig. 12(d) reflects the piggyback marking mode switching frequency in AOPM. We observe that as the path length increases, AOPM more frequently utilizes the one-hop piggyback marking opportunities to deliver message fragments. However, as shown in Fig. 12(c), SWAP incurs more than 2.3 times the number of enqueue operations than AOPM when the path length is 5, and the gap between them becomes larger with increasing path length.

2) Impact of Number of Message Fragments: Next, we investigate the performance of our design when varying the number of fragments per traceback message, which is determined by different message content encoding schemes. The path length of the flow of interest is set to 25 and $T_{ms}$ is set to 1s by default.

Fig. 13(a) plots the fragment delivery ratio as the number of message fragments varies from 5 to 20. The larger the number of message fragments per message, the more carriers are needed, and thus the traceback process lasts longer time. Consequently, it is more likely a fragment is dropped due to the buffering deadline expiration. Therefore, there is more or less decrease of delivery ratio in all schemes as the number of message fragments increases. However, AOPM maintains a high delivery ratio above 90% regardless the change of the number of fragments per message.

Fig. 13(b) shows the normalized traceback delay for different schemes. When a message is split only into 5 message fragments, all schemes show similar performance in term of the normalized traceback delay. However, as the number of message fragments increase, AOPM obviously outperforms the other schemes. For example, compared with SWAP when the number of fragments is 20, AOPM reduces the traceback delay by 50% on average.

The results of normalized enqueue operation overhead are depicted in Fig. 13(c). As a message being split into more message fragments, the enqueue operation overhead also increases for all schemes. When the number of fragments is 5, SWAP shows a 4 times higher enqueue operation overhead than AOPM. We observe that the enqueue operation overhead in AOPM becomes 60% of the overhead in SWAP when the number of fragments is increased to 20. Fig. 13(d) explains the underlying reason behind the results in Fig. 13(c). As indicated in the figure, the larger the number of fragments per message, the more one-hop carriers are used for message fragment delivery in AOPM. As a result, the enqueue operation overhead in AOPM also significantly increases.

3) Impact of Mode Switching Threshold: We now investigate the tradeoff between traceback delivery delay and enqueue operation overhead in AOPM. The path length of the flow of interest is set to 25 and the number of fragments per message is set to 10 by default.

Fig. 14 compares the traceback performance when varying the piggyback marking mode switching threshold $T_{ms}$ from 0.5s to 1.5s. In this comparison, the fragment delivery ratios of both AOPM and SWAP are higher than 90%. Note that $T_{ms}$ is a system parameter in AOPM, and thus has no impact on other schemes.

From Fig. 14(a), the traceback delay of AOPM increases with increasing the mode switching threshold when there are plenty of one-hop piggyback marking opportunities. In
the mean time, Fig. 14(b) shows that the enqueue operation overhead is controlled by adjusting the mode switching threshold in AOPM. If $T_{ms}$ is set a value larger than the fragment buffering expiration bound $T_d$, it means the piggyback marking mode is disabled.

From the analysis and evaluation results, we know that direct piggyback marking and one-hop piggyback marking have different benefits and are complementary to each other for traceback message delivery in MBT. OPM achieves a comparable performance regarding the traceback completion delay to AOPM when there are plenty of direct piggyback marking opportunities (as shown in Table III). While the main benefit of one-hop piggyback marking is not to reduce the traceback delay, but to improve the robustness of traceback by avoiding message fragments being dropped from buffer especially when there are insufficient direct piggyback marking opportunities in the network. Note that the simulation results are meaningful even when traffic rates at routers are much higher than our simulation settings (e.g., for typical Internet routers, the link speed can reach or exceed 100 Gbit/s), since the performance comparison of different message fragment delivery schemes mainly depend on the traffic distributions (e.g., the ratio of internal-flow to external-flow) in the network.

In FMBT, traceback messages will be sent to one or more traceback servers, rather than victims. Consequently, delivery of traceback messages in FMBT is independent of the traffic volumes of flows being traced. If there is no packet going to the traceback server, then piggybacking will not be of use. In this case, dedicated communication channel (with aggregation at each hop) will be needed. When there are sufficient background traffic flows, traceback messages can be efficiently delivered to the designated traceback server. For example, traceback servers may be deployed by ISPs in their internal data centers, and traffic going to the data centers (over 10 GB per server per day) reported in [51]) can be potentially exploited for message delivery in FMBT.

**IX. CONCLUSION**

In this work, we proposed opportunistic piggyback marking, a novel traceback acceleration mechanism for IP traceback. The main idea is to exploit free ride opportunities for expedited and robust delivery of traceback message fragments to end-hosts. Based on this idea, we designed a trigger-based IP traceback approach, which supports the traceback of individual packets. We then provided a theoretical analysis of marking-based traceback, and showed the potential of opportunistic piggyback marking. We also presented a flexible marking-based traceback (FMBT) framework, which meets several favorable objectives that previous individual traceback schemes failed to satisfy simultaneously. Comprehensive performance comparisons demonstrated the effectiveness and efficiency of our design for IP traceback. As for our future work, we would like to investigate counter-measures to mitigate the problem of compromised routers in marking-based IP traceback, address the robustness of message delivery in FMBT, and implement OPM/AOPM on a real network environment.

**ACKNOWLEDGEMENT**

This material is based on research work supported by the Singapore National Research Foundation under NCR Award No. NRF2014NCR-NCR001-034.

**REFERENCES**

C. Gong and K. Sarac, “Toward a practical packet marking approach for
D. Dean, M. Franklin, and A. Stubblefield, “An algebraic approach to
M. Adler, “Trade-offs in probabilistic packet marking for IP traceback,”
T. Peng, C. Leckie, and K. Ramamohanarao, “Adjusted probabilistic
H. Tian and J. Bi, “An incrementally deployable flow-based scheme for
T. Law, J. Lui, and D. Yau, “You can run, but you can’t hide: an
D. M. Divakaran, “A spike-detecting aqm to deal with elephants,”
C. Gong and K. Sarac, “Toward a practical packet marking approach for
Long Cheng is a Research Scientist at Institute for Infocomm Research (I²R), Singapore. He received his Ph.D. degree in Computer Science from the State Key Lab of Network and Switching Technology, Beijing University of Posts and Telecommunications, China in 2012. Before joining I²R, he worked as a Research Fellow at Singapore University of Technology and Design from March 2012 to Aug 2014. During the period from Jan 2009 to Sept 2011, he was a research assistant in the Hong Kong Polytechnic University and a visiting student in the University of Texas at Arlington, USA. Dr. Cheng is the author and co-author of over 50 papers in international journals and conferences, including ToN, TPDS, TII, SenSys, INFOCOM, ICNP, SECON, IWQoS, MASS and ICPP. He received the Best Paper Award of IEEE WCNC 2013 and Erasmus Mundus Scholar Award in 2014. His research interests include network security and forensics, wireless sensor networks, cyber-physical systems, mobile and pervasive computing. He is a member of ACM and IEEE.
Dinmil Mon Divakaran is a Research Scientist at the Cyber Security & Intelligence Department in the Institute for Infocomm Research (I²R), Singapore. Prior to this, for two years he was a Research Fellow at the Department of Electrical and Computer Engineering in the National University of Singapore (NUS). He has also worked as an Assistant Professor in the School of Computing and Electrical Engineering at the Indian Institute of Technology (IIT) Mandi. He carried out his PhD at the INRIA team in ENS Lyon, France. He holds a Master degree from IIT Madras, India. His research works revolve around applications of game theory, queueing theory and probabilistic models, as well as the study of optimization problems and design of heuristics, all in the broad area of computer networks. He is keenly interested in the study of architectures and protocols in the context of network security and QoS delivery.
Whee Yong Lim is the Deputy Lab Head for the Security Intelligence Lab, in Cyber Security & Intelligence R&D Department at the Institute for Infocomm Research (I²R), A*STAR, Singapore. His research interest includes predictive intelligence, text analysis, object recognition and machine learning.
Vrizlyn L. L. Thing leads the Cyber Security & Intelligence (CSI) R&D Department at the Institute for Infocomm Research, A*STAR, Singapore. The department focuses on digital forensics, cybercrime analysis, cyber security & intelligence, and mobile security research and technology development. She is also an Adjunct Associate Professor at the National University of Singapore (School of Computing), and the Singapore Management University (School of Information Systems). She has over 11 years of experience in security and forensics R&D experience in various aspects of cyber security forensics & intelligence. Her research draws on her multidisciplinary background in computer science (Ph.D. from Imperial College London, United Kingdom), and electrical, electronics, computer and communications engineering (B.Eng. and M.Eng by Research from Nanyang Technological University, Singapore). During her career, she has taken on various advisory and consultancy roles for some of the leading cyber security firms and the Lead Scientist of collaborative projects with industry partners such as MNCs, SMEs and the government agencies. More information about her work can be found at www1.i2r.a-star.edu.sg/~vriz.