Detecting flooding attack and accommodating burst traffic in delay-tolerant networks

Pham, Thi Ngoc Diep; Yeo, Chai Kiat; Yanai, Naoto; Fujiwara, Toru

2018

Pham, T. N. D., Yeo, C. K., Yanai, N., & Fujiwara, T. (2018). Detecting flooding attack and accommodating burst traffic in delay-tolerant networks. *IEEE Transactions on Vehicular Technology*, 67(1), 795-808. doi:10.1109/TVT.2017.2748345

<https://hdl.handle.net/10356/104140>

<https://doi.org/10.1109/TVT.2017.2748345>

© 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at:
<https://doi.org/10.1109/TVT.2017.2748345>.

Downloaded on 26 Apr 2023 19:02:23 SGT

Detecting Flooding Attack and Accommodating Burst Traffic in Delay Tolerant Networks

Thi Ngoc Diep Pham, Chai Kiat Yeo, Naoto Yanai, Toru Fujiwara

Abstract—Delay Tolerant Network (DTN) is developed to cope with intermittent connectivity and long delay in wireless networks. Due to limited connectivity, DTN is vulnerable to flooding attack in which malicious nodes flood the network with superfluous data to deplete the network resources. Existing works mitigate internal flooding attacks by rate-limit to constrain the number of messages that nodes can generate per time slot. However, rate-limit cannot flexibly accommodate burst traffic in which nodes may have sending demands higher than the rate-limit for a short period. In this paper, we propose FDER to detect flooding attack and yet allow legitimate burst traffic simultaneously. Nodes exchange their histories of encounter records (ER) which record the sent messages during their previous encounters. The ER history is used to infer a node's new message transmission rate over time and the number of forwarded replicas per message. The adversary nodes that send too many messages or replicas can thus be detected. Since ERs serve as useful tools for monitoring the sending behavior of nodes over a long time period, FDER could detect the burst traffic violation efficiently. We also design FP - a fairness forwarding policy to ensure fairness in the delivery performance fairness between nodes with normal traffic and those with burst traffic. FP uses ER information to observe nodes' rate of new message generation and adjust their forwarding priorities accordingly. Simulation results show that FDER can detect flooding attack at a higher accuracy and a lower delay compared to state-of-the-art scheme with affordable overhead. Moreover, FP could mitigate the smart flooding attack and still provide the performance fairness to support bursty traffic scenario.

Index Terms—flooding attack, encounter record, DTN

I. INTRODUCTION

Delay Tolerant Network (DTN) [1]–[3], which is characterized by intermittent connection, long transmission delay and lack of end-to-end path, is applied in many network scenarios, including sensor deployment in harsh environments [4], [5], disaster areas [6], and vehicular communication [7]. In DTN, messages are routed to the destination in a “store-and-forward” fashion: each message is first stored in the permanent storage of the source node or relay nodes and then forwarded opportunistically to optimal next hops or the destination node. To increase the delivery rate, most routing protocols allow data to be replicated and propagated along different paths to the destination. However, malicious nodes may utilize this feature to launch flooding attack. By sending

out large number of new messages, the adversaries deplete the scarce resources for legitimate data, thus severely degrading the delivery performance of benign nodes.

While flooding attack is well addressed in wired networks [8], [9] and mobile ad hoc networks (MANET) [10]–[13], these solutions cannot be directly applied to protect mobile DTN whose connectivity is transient and not available all the time. Many solutions have been proposed to prevent flooding attack in DTN. Ansa et. al. [14] designed a lightweight authentication scheme. Lee et. al. [15] regulated the dropping policy for node's message queue. However, [14], [15] cannot detect internal flooding attackers. Other works like [16], [17] limited the nodes' message generation rates and detected internal adversaries that exceed the rate limits. However, the rate limit approach is not flexible enough for networks with unpredictable transmission demands.

In this paper, we propose Flooding Detection based on Encounter Record (FDER) which aims at detecting flooding attack in DTN without imposing strict rate limit on a node's message generation rate. Instead, we introduce burst-limit policy to restrict flooding attack while flexibly accommodating a node's transmission demand. Specifically, each node's normal traffic pattern is still constrained by a rate limit but the node is allowed to have a small and short burst of new messages which exceeds the rate limit. However, if the node generates a large burst continuously for a long period, it is deemed as a flooding attacker and will be detected by FDER.

To achieve the above goal, each node needs to obtain the number of messages/replicas that other nodes have sent in the past. This is challenging in DTN since the data are routed opportunistically and do not follow predefined paths. Thus no single node can monitor all the sending activities of other nodes and there is no central server to control the network traffic either. Our solution is to let each node record what it has sent in each encounter with another node, i.e. when two nodes make a contact, they create a new encounter record (ER) [18] to record the list of messages they exchange during the encounter. ER is protected by the signatures of both sides so that its content cannot be forged. In additional, during the contact, two nodes exchange several most recent ERs so that they can judge if the neighbor's sending history violates the burst-limit policy. In this way, nodes can detect the malicious nodes in a distributed manner.

Enabling burst transmission raises the concern that the nodes sending burst traffic could occupy more network resources than those who comply with the rate limit. To ensure fairness among nodes with different traffic patterns and to mitigate the effect of the flooding misbehavior before it is

T.N.D. Pham, C.K. Yeo are with the School of Computer Science and Engineering, Nanyang Technological University, N. Yanai and T. Fujiwara are with the Graduate School of Information Science and Technology, Osaka University.

E-mail: pham0069@e.ntu.edu.sg, asckyeo@ntu.edu.sg, yanai@ist.osaka-u.ac.jp, fujiwara@ist.osaka-u.ac.jp

Manuscript received

detected, we design Fairness Policy (FP) to lower the forwarding priority of messages which are generated by nodes with high generation rates. This additional feature can be easily incorporated into existing DTN routing protocols.

To validate our scheme, we implement extensive simulations in ONE simulator. Simulation results show that FDER achieves high accuracy and low delay in detecting flooding attacks for different routing protocols. FDER also outperforms the state-of-the-art scheme [17] in detecting a violation of the burst limit. FP is shown to ensure the delivery performance of nodes with normal traffic.

The rest of the paper is organized as follows. Section II defines the flooding attack, states the assumptions and provides an overview of FDER. Section III describes the flooding attack detection mechanism of FDER. Section IV presents the fair queueing and forwarding mechanisms to mitigate the effects of the attack. The simulation results and evaluations are shown in Section V. Section VI summarizes the existing solutions against flooding attack. Section VII concludes the paper.

II. OVERVIEW

A. Background and Motivation

Owing to intermittent connectivity, it is hard to establish end-to-end path from the source to the destination of a message. Instead, DTN's routing strategy is "store-and-forward": nodes first store the message and then forward it to the appropriate next hop or the destination opportunistically. To increase the delivery success, most routing protocols in DTN allow messages to be replicated and disseminated in different paths. In some routing protocols such as Prophet [19] and Maxprop [20], there is no limit on the replica number for each message. Specifically, after a node relays a message to another node, it still keeps that message to forward in later contact opportunities. Other routing schemes such as Spray and Wait [21] aim at limiting the total number of replicas for each distinct message in the network. Thus, the source and relay nodes of the message can forward it for a limited number of times which is defined by the routing protocol. We refer to the former routing strategy as propagation routing and the latter as quota-based routing.

Utilizing the multi-copy feature of DTN routing protocols, malicious nodes may launch the flooding attack as follows. Malicious nodes generate and send too many unnecessary messages, which in turn would be replicated into a huge amount of bogus data over the network. Moreover, in quota-based routing, malicious nodes may replicate each of their messages for a number of times higher than the defined threshold. We refer to the former attack strategy as message flooding and the latter as replica flooding. Both flooding strategies spawn invalid data that contend for network resources with legitimate data and ultimately reduce the delivery rate of normal nodes' messages. This motivates us to design a scheme to protect DTN against flooding attack.

B. Defense lines against flooding attack

In this section, we state the defense mechanisms used to mitigate the flooding attack described above. The first

line of defense is authentication. Legitimate nodes own valid keys and sign their own messages so that other nodes can authenticate the source nodes of the messages. Nodes only keep and forward messages generated by internal nodes. Thus authentication deters external flooding attackers that inject unauthorized data into the network. However, authentication cannot detect internal adversaries that deliberately flood messages or replicas.

The second line of defense is the rate limit introduced in [17]. It is supposed that the time is divided into equal fixed slots of duration T , starting from 0, T , $2T$ etc. Each node has a rate limit M which is the maximum number of new messages that it is allowed to generate and send per time interval T . Rate limit mitigates the message flooding attack but the fixed rate constrains nodes from sending messages on demand. Flexible rate is possible [17] by allowing nodes to request the authority for new rate-limit permissions when their sending demands change. However, this solution is infeasible if the connectivity to the authority is limited or costly especially if the nodes' sending demands are unpredictable.

Thus we introduce burst limit as an alternative to rate limit that flexibly accommodates nodes' short-period burst traffic. Specifically, nodes are allowed to exceed the rate limit by at most K messages per slot and have at most Q consecutive burst slots. An interval, in which a node sends more than M messages but less than or equal to $M + K$ messages, is called a valid burst slot. In this way, the excess messages in the valid burst slot still get some chance of being delivered if the network resources are sufficient. The nodes that violate these rules are deemed as attackers.

The third line of defense is replica limit which is the maximum number of copies for each message that a node is allowed to forward in the network. The replica limit is normally defined by the routing protocol. For example, Spray and Wait [21] with replica limit L only allows the source node of a message to forward that message for at most L times. This aims at mitigating the replica flooding attack.

C. System model

1) *Network model*: The network consists of mobile nodes communicating in ad-hoc mode using wireless technology such as Wi-Fi. Each node owns a unique identifier. The network is loosely time synchronized such that if the time is divided into time slots of equal length, any two nodes are in the same time slot at any one time.

We assume authentication is enabled by identity-based cryptography (IBC) which involves a key generator center to generate a valid private key for each node from the node's identifier. Due to no requirement of cumbersome infrastructures and light overhead, IBC is suitable for mobile ad-hoc network like DTN [22].

We adopt the burst limit and replica limit mentioned above as the approach to mitigate internal flooding attacks in DTN.

2) *Attack model*: Attackers can flood messages or replicas or both. During flooding, an attacker either exceeds its rate-limit M by more than K messages per interval or continuously exceeds its rate-limit M for more than consecutive Q intervals.

When flooding replicas, an attacker exceeds the replica limit L which is defined by the routing protocol. Fig. 1 summarizes the factors involved in different attacks. It is noted that since there is no replica limit in propagation routing protocols, we only consider replica flooding attack in quota-based routing protocols.

Fig. 1: Factors of flooding attack.

D. Basic ideas

Adopting rate/burst and replica limit to mitigate flooding attacks involves the evaluation if a node has generated more messages or forwarded more replicas than the thresholds. However, monitoring how many messages and replicas that a node has forwarded is non-trivial in DTN since data are routed opportunistically and no single node can keep track of all the sending activities of other nodes.

To resolve this issue, we introduce the message header and encounter record (ER) as the tools for nodes to monitor how many messages and replicas other nodes have forwarded in a distributed manner. Specifically, the header of a message is created by the source node to specify the order in each time slot in which the message is sent for the first time, i.e. message count. The ER records the messages that a node has sent and received during an encounter with another node by specifying the message headers. For each sent message recorded in ER, the sending node also counts the number of times it has replicated that message, i.e. replica count. Each node keeps its most recent ERs and presents them to other nodes it encounters so that these nodes can evaluate its transmission behavior. An attacker that flood messages or replicas will have its message counts or replica counts higher than the given limits. Thus by looking at the adversary's ERs, other nodes can detect the flooding attack.

III. MISBEHAVIOR DETECTION SCHEME

In Section III-A and III-B, we present the construction of message header and ER and how they are used to detect message and replica flooding attacks. The message headers and ERs reflect the transmission histories of the nodes, thereby enabling the monitoring of transmissions which exceed the given limits. The adversaries may hide their flooding attacks by manipulating their ERs or the actual fields in the ERs. We describe these misbehaviors in details and propose the corresponding detection procedures in Section III-C, III-D and III-E.

A. Message model

The time is divided into fixed time slots of length T , starting from 0, T , $2T$, etc. When a node has something new to send, it first generates the message content. When it has a chance to send the message for the first time, it creates the message header to record the source and destination nodes, the current time and the message count. The message count reflects the order in which this message is sent for the first time in this time slot. Finally, the source node signs off the message content and it is concatenated with the message header. The signature ensures the message is created by a legitimate node and protects the content and header from being modified by intermediate nodes.

For a new message m , the source node creates the message header H_m which is formatted as follows:

$$H_m = \langle S, D, ts, c \rangle$$

$$m^* = \langle m, H_m, sig \rangle$$

where S and D are the respective identifiers of m 's source and destination nodes, ts is the current time slot when S first sends m and c is the message count of m (i.e. m is the c^{th} new message that S has created and sent out in ts), sig is the source node's signature over the content and header, m^* is the concatenation of message m , header H_m and signature sig .

The message header and signature are always forwarded together with the message content to later hops (i.e. m^* is replicated in relay nodes). When an intermediate node receives m^* , it first verifies the signature and checks if c exceeds $M + K$. If the signature is invalid, it blacklists the previous hop since it forwards a corrupted message. If the signature is valid but c is larger than $M + K$, it blacklists the source node of m^* and alerts other nodes about the blacklisted node. Otherwise, it keeps m^* in its local storage.

1) *Alert on the adversaries*: Once an adversary is detected, the alert procedure is implemented as follows. The node that has authenticated evidences to prove the misbehavior of the adversary will add it to the blacklist and keep the evidences. From then on, when it encounters other nodes, it forwards the blacklist to the peers. If the peers have not blacklisted any listed adversary, the peers will request the node to send the corresponding evidences. The peers first verify the evidences; if they are valid, they blacklist the adversary and help in further alerting the network. Since the evidences are authenticated, badmouthing attack, in which a node accuses a normal node as an adversary, is impossible.

2) *Batch authentication*: Each node is required to verify the signatures of all the received messages, which will incur high overhead for a large number of messages. To reduce the authentication overhead, we adopt the batch authentication technique that allows a node to verify multiple messages which are signed by multiple users in a fixed number of operations which is independent of the number of messages.

In particular, we use batch verification scheme of IBC-based signatures [23] which is enabled by the bilinear feature of pairings. The algorithm details are summarized in Fig. 2. The verification method requires several types of operations including pairing computation (i.e. $e(x, y)$), hashing (i.e. H_1, H_2), scalar multiplication, elliptic curve addition. Since pairing

is much less efficient than other operations, the verification efficiency thus depends on the number of pairings. As seen in Fig. 2, if η messages are individually verified, the number of pairings is $2 \times \eta$. Meanwhile using the batch verification, η messages can be verified in 2 pairing operations only. Thus the verification cost is significantly reduced, especially when the number of messages η is high.

B. ER model

1) *Generating ER*: When two nodes encounter each other, they create an ER that records their identities, the ER sequence numbers assigned by them, the encounter timestamp and the list of exchanged messages. For each message that a node forwards in this encounter, the node records the message header information and the number of times it has replicated this message so far in the ER. Finally, both nodes sign off the ER content to prevent each side from modifying the record and store this authenticated ER in their local storage. Owing to the limitation of storage, each node only keeps a window of W most recent ERs.

When a node i encounters another node j , node i generates ER_i :

$$\begin{aligned} ER_i &= \langle ID_i, ID_j, sn_i, sn_j, t, SL_{ij}, RL_{ij} \rangle \\ SL_{ij} &= \{MR_{ij}^m \mid i \text{ sends message } m \text{ to } j\} \\ RL_{ij} &= \{MR_{ji}^m \mid j \text{ sends message } m \text{ to } i\} \\ ER_i^* &= ER_i, sig_i, sig_j \end{aligned}$$

where ID_i, ID_j are their respective identifications, sn_i, sn_j are the sequence numbers assigned to the ERs by nodes i and j respectively; t is the encounter time; sig_i, sig_j are their signatures over ER_i using their respective private keys. A node assigns its new ER with the sequence number of its latest ER incremented by 1. SL_{ij} and RL_{ij} denote the lists of message records MR that node i sends to and receives from node j . MR_{ij}^m , denoting the record of message m that node i sends to node j , comprises the fields shown as follows:

$$MR_{ij}^m = \langle H_m, r_{im} \rangle$$

where H_m denotes the message header of m as mentioned in Section III-A. r_{im} indicates the replica count, i.e. this is the r_{im}^{th} time that the sender i (either as intermediate or source node of m) has forwarded m . It is noted that node i cannot modify H_m to badmouth the source node (for example, by claiming a very large MSG_m) since j can verify the message header which is protected by the source node's signature. Node i stores the authenticated ER_i^* in its local storage.

2) *Detecting flooding attack with ER*: When two nodes i and j encounter each other, they exchange their most recent ER windows to evaluate each other's sending behavior. Node i first verifies the signatures of the ERs submitted by node j . Batch authentication mentioned in Section III-A2 is used to accelerate the ER authentication. If node j 's ER window passes the verification, node i proceeds to check if j exceeds the burst limit as follows. From node j 's ER history, node i traverses through the lists of sent messages of node j and extracts the list SSL_j recording the times node j sends its own messages:

$$SSL_j = \{MR_{jx}^m \mid \text{the source node of } m \text{ is } j \text{ and } j \text{ sent message } m \text{ to some node } x \text{ in the given ERs of node } j\}$$

Based on SSL_j , node i can infer the maximum message count of node j over the past time slots. Node i checks if any maximum message count exceeds $M + K$ or if there exists $Q + 1$ consecutive slots in which all the maximum message counts exceed M . Furthermore, if quota-based routing protocol is used, node i checks each message record in SSL_j to see if any replica count exceeds L .

If node i detects that node j violates burst or replica limit, it will blacklist j and alert other nodes about the adversary j . Node j also performs similar authentication and evaluation process on node i 's ER.

3) *Attack model against ER*: To avoid the flooding attack detection, an attacker may launch attacks against ER

Setup	- Let G_1, G_2 be two cyclic groups of prime order q, P be the generator of G_1, - Let $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear pairing which satisfies the following 3 conditions: - <i>bilinear</i>: $\forall x, y \in G_1, a, b \in \mathbb{Z}_q$, we have $e(a^x, b^y) = e(x, y)^{ab}$, - <i>non-degenerate</i>: $e(P, P) \neq 1$ - <i>computable</i>: $\exists$ efficient algorithm to compute $e(x, y) \forall x, y \in G_1$ - Pick a random $s \in \mathbb{Z}_q$ and compute $P_{pub} = P^s$ - Choose two hash functions $H_1 : \{0, 1\}^* \rightarrow G_1$ and $H_2 : \{0, 1\}^* \times G_1 \rightarrow \mathbb{Z}_q$ - The system parameter is (P, P_{pub}, H_1, H_2). The master key is s.
Extract	Given an identity ID , compute $Q_{ID} = H_1(ID)$ and $D_{ID} = Q_{ID}^s$. The private key is D_{ID} .
Signing	Given a private key D_{ID} and a message m , pick a random $r \in \mathbb{Z}_q$ and compute $U = Q_{ID}^r$, $h = H_2(m, U)$, $V = D_{ID}^{h+r}$. The signature is $\sigma = (U, V)$.
Verification	Given the signature is $\sigma = (U, V)$ of message m for an identity ID , compute $h = H_2(m, U)$. Accept the signature if $e(P, V) = e(P_{pub}, U \cdot Q_{ID}^h)$.
Batch verification	Given η signatures $(ID_1, m_1, U_1, h_1, V_1) \dots (ID_\eta, m_\eta, U_\eta, h_\eta, V_\eta)$, compute $Q_i = H_1(ID_i)$ and $h_i = H_2(m_i, U_i)$. Accept the signatures if $e(P, \prod_{i=1}^\eta V_i) = e(P_{pub}, \prod_{i=1}^\eta U_i \cdot Q_i^{h_i})$.

Fig. 2: Identity-based cryptography signatures and batch verification

- 1) manipulating ER: the adversary deletes unfavorable ER from its history
- 2) manipulating message count: the adversary declares wrong message counts in ER
- 3) manipulating replica count: the adversary declares wrong replica counts in ER

We analyze each of these misbehavior respectively in Sections III-C, III-D and III-E. It is assumed that there is no collusion attack in which the adversaries cooperate with one another to create fake ERs or incorrect ER information. This issue will be left to the future work.

C. Detecting ER manipulation

1) *Misbehavior*: An attacker may delete unfavorable ERs (i.e. those revealing its misbehavior) from the ER history to obtain an ER window beneficial for itself to present to other nodes. For example as in Fig. 3, the adversary's actual ER windows are *A*, *B*, *C*, *D* with the respective timestamps of 10, 20, 30, 40 and respective sequence number (assigned by attacker) of 1, 2, 3, 4. If it wants to delete the ER *C* with the sequence number 3, it can use one of the following three manipulation strategies:

- 1) *Skip ER sequence number*: The adversary deletes ER *C* and does not use the sequence number 3 in any other ER.
- 2) *Reuse ER sequence number*: The adversary deletes ER *C* and assigns the sequence number 3 to ER *E* that occurs at the later timestamp of 50.
- 3) *Hold back unfavorable ERs*: The adversary holds its sequence number at 3 to generate different ERs i.e. *C* and *D*. After achieving a favorable ER, it will increment the sequence number for later new ERs.

Fig. 3: ER manipulation strategies.

2) *Observation*: According to the rule of creating ER in Section III-B1, a series of consecutive well-behaved ERs have sequential sequence numbers. Besides, ER with higher sequence number has a bigger timestamp. However, adversaries using strategies 1 and 2 described above violate either rules. In the first strategy, the new ER history has the series of sequence numbers = 1, 2, 4 which is not sequential. In the second strategy, the new ER history has the series of time stamps = 10, 20, 50, 40 which is not monotonically increasing.

When the manipulated ER window is presented to a neighbor node in later encounter, the misbehavior is easily detected.

When adversaries use strategy 3, the manipulated ERs history is *A*, *B*, *D* which abide by both sequence number and timestamp rules. Thus the neighbor cannot discover the manipulation at all if it only depends on the ER window presented by the malicious node. However, the misbehavior leads to inconsistency in that at least two different records *C* and *D* have been assigned with the same sequence number by the adversary.

Fig. 4 illustrates how to detect the adversary *q* that uses the same sequence number 3 to create ERs *C* and *D* with two different nodes *i* and *j*. If nodes *i* and *j* meet each other and exchange their ERs, they will directly discover that node *q* has manipulated its ERs. However, if node *q* selectively manipulates the ERs with two nodes *i* and *j* that rarely meet, the possibility that *q* gets detected is much lower. To deal with this strategy, we need another node *k* who meets both *i* and *j* and stores their ERs *C* and *D* locally. Upon encountering *i* and receiving its ERs, *k* notes and stores ER *C* locally. If later on, *k* also encounters *j*, it can check ER *D* of *j* against ER *C* and discover that *q* has assigned the sequence number 3 to different ERs.

Fig. 4: Detecting ER manipulation.

This detection process requires node *k* to store several ERs of node *i* it encounters. To achieve efficient detection at low cost, we propose the sampling storage process as follows. Each node is required to maintain a global ER database (GERDB) in its local storage. When the node receives an ER history of a neighbor, it randomly chooses a few ERs of the neighbor and stores them in its GERDB. Since storing all the received ERs is expensive, we only let each node sample a small number α of the received ERs and limit the maximum number of ERs in GERDB to β . When GERDB's capacity is full and new ERs are added, the oldest timestamp ER is deleted to make room for the new one. As a result, GERDB provides each node with a partial global view of sequence number assignment over the network.

3) *Detection*: When node *i* encounters and receives the ER window of node *j*, node *i* first examines if node *j*'s ERs have their sequence numbers assigned by node *j* follow the sequential order and be consistent with the timestamp order. Next node *i* cross checks node *j*'s ERs against the ERs in its GERDB to inspect if any node has assigned the same sequence number to different ERs. If node *i* detects any inconsistencies due to any node, it will blacklist that node and alert other

nodes.

D. Detecting message count manipulation

1) *Misbehavior*: Fig. 5 shows an example of an attacker that claims the wrong message count when it launches message flooding attack. The ER window size W is set to 4, the rate-limit M is set to 3 but the attacker has generated and sent out 4 distinct messages m_1, m_2, m_3 and m_4 in the current time slot 1. If it keeps the count authentic, the message header of m_4 reveals its violation of the rate limit. To avoid being detected easily, the attacker can claim the message count with a smaller value (such as 1) than the real value, i.e. reusing message count.

Fig. 5: Manipulating message count.

2) *Observation*: An adversary can use the same message count 1 for m_1 and m_4 to hide its flooding attack. However, when m_1 and m_4 are disseminated in the network, they will be deemed as the same message (despite their contents being different) since they have the same source node, message count and time slot. For existing routing protocols in DTN, upon encounter, nodes only exchange messages which they have not relayed by comparing their lists of stored messages. Thus the intermediate nodes that have already kept m_1 in their buffers will not accept m_4 and vice versa. The total number of replicas of m_1 and m_4 in the network is therefore restricted.

To increase the number of nodes to receive either m_1 or m_4 , the adversary q may assign m_4 a destination different from that of m_1 . However, this misbehavior leads to the inconsistency of destination nodes between two messages with the same source node, message count and timestamp. A node i could detect this misbehavior in two scenarios:

- Suppose node q sends m_1 and m_4 frequently such that the record of both messages are present in its ER window. When node q submits this ER window to an encountered node i , node i will observe the inconsistency in the destination.
- Suppose node i has already stored m_1 . Later, node i may discover about m_4 when it encounters a node whose ER history involves the record of m_4 . Another way for node i to observe m_4 is when it encounters a node that has already stored m_4 and they compare their lists of messages as part of the routing process. In either case, node i can detect the inconsistency in the destination.

3) *Detection*: When node i encounters node j and receives its ER window, node i checks if there exists any pair of message records in j 's ER history such that they have the same source node, same message count and same time slot used for sending but different destination nodes. Node i also checks the message records in node j 's ERs against the headers of messages that node i stores to inspect the inconsistency in the destination node. Upon detecting any inconsistency between a pair of message records or headers, node i blacklists the source node of that message and alerts the network.

E. Detecting replica count manipulation

1) *Misbehavior*: When an attacker launches replica flooding attack and violates the replica limit, it may reuse existing replica count of the same message to hide its misbehavior as shown in Fig. 6 (a). The ER window size W is set to 4, the replica limit L is set to 3 but the attacker has forwarded the message m for 4 times. If it honestly declares the replica count, the fourth count is easily detected as being higher than the replica limit. Another choice for the attacker is to claim a replica count of a smaller value (such as 1) than the actual value, i.e. reusing replica count.

(a) Reusing replica count immediately.

Fig. 6: Manipulating replica count.

2) *Observation*: For a normal node that declares the correct replica count, the series of replica counts for each message in increasing order of sent timestamp is sequential. Note that the sent timestamp of a replica is equivalent to the creation timestamp of the related ER. However, when the adversary reuses replica count to hide its misbehavior, it violates the above observation. As shown in Fig. 6 (a), if the adversary reuses the replica count 1 for message m in ER E , the ER window (B, C, D, E) reflects that the series of replica counts of m in increasing order of sent time, which is 2, 3, 1, is non-sequential.

To prevent the replica count reuse strategy from being detected easily, the attacker may flood replicas in a later time such that the inconsistent replica counts are not present in the same ER window. For example, in Fig. 6 (b), the adversary

sends superfluous replicas of message m in ERs X, Y, Z when ERs A, B, C which record the previous replication of m are out of its ER window. When the adversary presents its ER window, its misbehavior cannot be detected directly. However, it can be detected if there exists a node that knows about both ERs A and X and reports the inconsistency in the replica count in these ERs.

Fig. 7 illustrates the detection process of the adversary q that claims the same replica count 1 when it replicates the message m to two different nodes i and j in the respective ERs A and X . Node q is detected if nodes i and j meet each other, exchange ERs and check the replica count records in its ERs against the peer's ERs. Node q is also detected if node k meets both i and j and stores their ERs A and X locally. Specifically, upon encountering i and receiving its ERs, k notes and stores ER A locally. If later on, k also encounters j , it can check ER X of j against ER A and discovers that q has reused the replica count. To implement this detection, we make use of GERDB mentioned above to enable node k to sample node i 's ERs and opportunistically detect node q .

Fig. 7: Detecting inconsistent assignment of replica count.

3) *Detection*: When node i encounters node j and receives its ER window, node i extracts the replica counts for each message m that node j has sent and checks if this sequence is sequential. If the sequence is not sequential, node i blacklists node j and alerts the network. Node i also checks the replica count records in node j 's ERs against those in node i 's ER window and node i 's GERDB. If there exists any pair of different records that indicates the same node sends the same message and claims the same replica count, that node is blacklisted and the network is alerted by node i .

F. How ER window mitigates flooding attack

In this section, we will explain the role of ER window in mitigating the flooding attack by showing the attacker's trade off between its flooding capability and the rate of reusing replica and message counts.

Fig. 8 (a) shows the tradeoff for attackers in replica flooding. The window size W and replica limit L are set as 4 and 3 respectively. It is observed that the more replicas the adversary floods, the longer it needs to wait to reuse the replica and message count. If the attacker sends 3 replicas of message m in ERs A, B and C , it needs to wait until ER G (6 ERs from A) to reuse the replica count 1 for m . Meanwhile, if it sends only 1 replica in ER A , it can reduce the reuse waiting time

Fig. 8: Attacker's trade off between the number of flooded replicas and the rate of reusing count.

to ER E (4 ERs from A) but this will also reduce its flooding efficiency.

Similarly, Fig. 8 (b) shows the tradeoff for attackers in message flooding. The window size W and rate-limit M are set as 4 and 3 respectively. Suppose the attacker uses the same message count for m_1 and m_2 but different destination nodes. If the attacker sends 3 replicas of message m_1 in the first three ERs, it needs to wait 6 ERs before reusing the message count 1 for m_2 . In contrast, if it sends only 1 replica, it can reuse the message count earlier but misses the chance to disseminate more replicas of message m_1 .

From the example in Fig. 8, we can generalize that to reuse a message or replica count, the attacker needs to wait for at least W ERs to pass. Thus setting a larger value for the window size W can help reduce the count reuse rate of attackers. Note that even if attackers can reuse the counts without being detected immediately, its misbehavior could still be detected opportunistically as presented in Sections III-D and III-E. We evaluate how window size affects the detection performance by simulation later in Section V-F.

IV. FAIRNESS POLICY

Although burst-limit policy provides nodes with more flexible transmission, it brings about the concern whether the nodes that generate frequent high traffic occupy more network resources than those that generate lower traffic. To compensate the effect of burst traffic and to mitigate the effect of flooding attack when it is not detected yet, we design FP to decide the priority of messages to be sent when a forwarding opportunity arises and of deleting them from the message queue. The main principle of the policy is that if a node is

observed to send out many new messages recently, other nodes will give its messages less priority to be forwarded.

A. Updating the forwarding priority

We propose that each node maintains and updates a metric called the forwarding priority for other nodes in the network. By observing the ERs provided by neighbors and the receiving messages, a node learns the maximum message count per time slot as declared by other nodes. The node maintains the maximum counts of other nodes in the most recent slots and uses them to update the forwarding priorities of other nodes. If the maximum count of a node is below the rate limit, its forwarding priority is raised and vice versa.

Denote f_j^i as the forwarding priority that node i gives to the messages generated by node j . The range of f_j^i is between 0 and 1. It is initiated with a default value $F \in [0, 1]$. Node i maintains the message summary (MS) to record the maximum message counts of node j for the most recent R slots:

$$MS_j^i = \langle ID_j, f_j^i, c(G), \dots, c(G - R + 1) \rangle$$

where ID_j denotes the identifier of node j , $c(g)$ is the maximum message counts of messages generated by node j in the slot g ($\forall g = G, G-1 \dots G-R+1$) that are known to i , G is the current time slot. Node i can learn the message counts of messages generated by node j when (1) node i meets another node that presents ERs involving the messages generated by node j , (2) node i receives messages generated by node j upon opportunistic encounter with other nodes. As ERs and the messages are authenticated with signatures, the message count value is ensured to be authentic. The update can only occur if the new maximum count is larger than the current maximum count for the same time slot. If node i has no information about node j message history in a slot g , the maximum count $c(g)$ is set to -1 . Note that $c(g)$ update can be piggybacked on the misbehavior detection process presented in Section III, so little overhead is incurred.

When node i removes the oldest maximum message count $c(g)$ from MS_j^i , node i updates f_j^i as follows. If $c(g)$ is equal -1 , node i does nothing since it has no information about node j 's message history during this time slot. If $c(g)$ is equal or less than the rate limit M , node i increases the forwarding priority:

$$f_j^i = \max(1, f_j^i \times \delta_1)$$

where δ_1 is a constant >1 . Otherwise, node i decreases the forwarding priority proportionately to its rate of sending new messages:

$$f_j^i = f_j^i \times (M \div c(g))^{\delta_2}$$

where δ_2 is a constant >1 . Before $c(g)$ is deleted, node i checks if node j has r ($\leq Q$) consecutive outburst time slots (i.e. maximum count exceeds the rate limit M) starting from this slot g . If this is the case, node i further decreases f_j^i :

$$f_j^i = f_j^i \times (\delta_3)^r$$

where δ_3 is a constant <1 .

Besides, we consider fairness in the aspect of rewarding the effort of each node which helps to forward the messages of

other nodes. When node i meets node j , node i can calculate SO_j and SI_j which are the respective number of messages that node j sends for other nodes and for itself during recent encounters. To compensate for the forwarding effort of node j , node i can increase f_j^i if the ratio of $SO_j \div SI_j$ is larger than a certain threshold:

$$f_j^i = \max(1, f_j^i \times \delta_4)$$

where δ_4 is a constant >1 .

B. Fair forwarding and queueing policy

Based on the calculated forwarding priority, we design the message forwarding and queueing policies as follows. When node i encounters node j , they exchange messages with each other. Node i will forward the messages based on their source node and in the order of decreasing forwarding priority. When node i receives a message but its buffer runs out of space, it needs to delete one or several messages to make room for the new message. Node i will delete messages in the order of increasing forwarding priority of their source nodes.

It is noted that the maximum message counts recorded by node i may be not the actual maximum counts of node j in a certain time slot. Thus the local value of f_j^i maintained by node i might be higher than reality. However, with the opportunistic contact and exchange of ERs among nodes, it is highly possible for node i to obtain enough information to make an acceptable estimate of f_j^i . Even if node i overestimates f_j^i and gives high priority to node j 's messages, the forwarding of these messages could be limited in other nodes who have more information about j . Thus the superfluous messages of node j can be mitigated in the network.

The forwarding and queueing policies help to mitigate the impact of the malicious node's strategy of bursting the set threshold. Consider a malicious node that sends more than M but less than or equal to $M + K$ new messages per slot (i.e. valid burst slot) for consecutive Q slots, then sends less than M messages (i.e. rate-limit slot) for the next slot, and continues the pattern of alternating between burst and rate-limit traffic. In this way, the malicious node can bypass the detection of message flooding. However, since the burst is too frequent, its forwarding priority is often being reduced more than increased, leading to a very low forwarding priority. As a result, the messages of the malicious nodes can be efficiently limited.

C. Integrating fair policies with existing routing protocols

Existing routing protocols in DTN normally focus on raising the average delivery ratio of the whole network but do not consider the fairness in the delivery performance among nodes. Thus their forwarding strategies is either to only spread messages not found in the next hop nodes to conserve bandwidth or to give higher priorities to messages that have higher chance of being delivered to the destination with the help of the encountered node. For example, in Epidemic and Spray and Wait, a node simply forwards all the messages which an encountered node does not have in its buffers yet. In Prophet and Maxprop, a node forwards a message to an encountered

peer if the peer has a higher possibility to deliver the message to its destination node.

We can integrate the forwarding and queueing policies above into existing routing protocols as follows. For Epidemic and Spray and Wait, each node simply adopts the forwarding policy for the messages to be forwarded to the peer. That means the messages generated by nodes with high forwarding priority are forwarded first. For Prophet and Maxprop, denote d as the metric used to measure the routing capability of the peer for each message. Denote f as the forwarding priority a node gives to the source node of that message. The messages will be forwarded in the increasing order of $f \times d$.

V. PERFORMANCE EVALUATIONS

A. Simulation setup

We simulate a sparse mobile network with a small number of nodes to achieve a scenario suitable for DTN. The general simulation setting is as follows. The simulation is implemented in ONE simulator [24]. We have 40 nodes with a transmission range of 100 meters. They travel over an area of 4500m x 3400m, at speeds of 10-50 km/h, using the Shortest Path Map Based Movement Model available in ONE to simulate the movement of vehicles on the city streets. The simulation time is 43200s or 12 hours. The number of attackers is 12. The window size W is 20. This setting is applied to all the following experiments by default. For each experiment with distinct setting, the simulation is run for 10 times using random seeds and the average of the measured metrics is recorded.

We assess the detection performance of FDER in detecting the three types of misbehaviors: ER manipulation, message flooding and replica flooding. We use the following metrics for the evaluation purpose:

- Detection accuracy: percentage of malicious nodes that can be detected by normal nodes
- Detection delay: the average time taken to detect attackers among those normal nodes that can successfully detect the misbehavior.
- Wasted transmissions percentage: wasted transmission is defined as the normal node's transmission (relay or delivery) of a message generated by an attacker. Wasted transmission ratio is the ratio between the number of wasted transmissions and overall number of transmissions.

B. Detecting ER manipulation misbehavior

Attackers hold back unfavorable ERs to hide their dropping behaviors, which is the third strategy described in Section III-C2. We consider this manipulation strategy because it is the most challenging to detect among the three strategies. We define ER manipulation ratio (EMR) as the ratio of the number of manipulated ERs over all the ERs for a malicious node. We vary the attackers' EMR from 0.01 to 0.2. The parameters α and β are set as 3 and 20 respectively.

Fig. 9 shows the detection rate and delay of normal nodes to detect attackers. We find that most of the malicious nodes can be accurately detected by all normal nodes when EMR is as

high as 0.04. The detection delay reduces and the detection rate increases as EMR increases. This means the more frequently the attackers manipulate the ERs, the faster and more probable they can be discovered.

Fig. 9: ER manipulation detection time and accuracy.

C. Detecting message flooding

In this section, we first evaluate the detection of message flooding when attackers reuse message count. The routing protocols used are Maxprop and Spray and Wait. The time interval T is one hour. The rate limit M is 6 messages per T . The number of excess messages that each attacker sends per T is varied from 1 to 5. We compare the detection performance of FDER with Li's scheme [17] in the same settings.

Fig. 10 plots the performances of FDER and Li's schemes in detecting the message count reuse. When attackers send more superfluous messages per time slot, the detection delay reduces and the detection accuracy increases. As a result, the wasted transmission ratio decreases when attackers flood more messages. This means both FDER and Li's schemes can successfully control the malicious nodes' impact on the network. FDER performs slightly better than Li's scheme in the detection delay and wasted transmission ratio. This is because in FDER, all the sending activities of a node are compactly recorded in ERs and presented to other encountered nodes. Thus the inconsistency of message counts appearing in different encounters of the adversary could be detected very quickly. Meanwhile in Li's scheme, the message count records are randomly distributed to different nodes and the inconsistency detection can thus only occur opportunistically unlike FDER's approach using ERs.

Next we evaluate the detection of message flooding when attackers exceed the number of valid burst slots Q . The settings used in the routing protocol, M and T , are as above. Furthermore, K is fixed as 5, Q is varied from 1 to 5. Each attacker is set to send more than M but less than or equal to $M + K$ messages per time slot for more than Q consecutive time slots. Attackers declare the correct message count. Though Li's scheme is not designed specifically to enable burst-limit policy, we revise Li's scheme to detect the burst-limit violation for the purpose of comparison.

Fig. 11 shows the performances of FDER and Li's scheme in detecting the violation of threshold Q . The general trend to observe in both schemes is that when the threshold Q

Fig. 10: Performance of detecting message count reuse.

Fig. 11: Performance of detecting burst-limit violation.

increases, the detection delay and the wasted transmission ratio increases while the detection accuracy decreases. This is because when Q is larger, a node needs to collect the attackers' sending behavior over a longer time, which is more difficult and thus leads to the possibility that the node might not have enough information to conclude that it is indeed an attacker. It is seen that the detection accuracy of FDER is much higher than Li's scheme in this scenario. This proves the capability of ER to facilitate the monitoring of a node's sending behavior over multiple time slots.

D. Detecting replica flooding

We evaluate the detection of replica flooding when attackers reuse replica count. We use the quota-based routing protocol called Spray and Wait. The initial number of copies L for normal node is set to 5. The number of excess replicas per message of each attacker is varied from 1 to 5.

Fig. 12 plots the detection performances of FDER and Li's schemes with varying excess replica rate. When the number of superfluous replicas increases, the detection delay reduces and the detection accuracy increases. This is because the more replicas the attackers inject into the network, the higher is the possibility that they leave traces and get detected by the defence schemes. FDER's performance is higher than Li's scheme since the ER window in FDER helps the nodes to

quickly detect the inconsistency of replica counts assigned by the attackers over the different encounters.

E. Countering maximum burst attack with FP

We evaluate the performance of FP by integrating it with the routing protocol Prophet. We set each attacker to launch the maximum burst attack, i.e. it sends $M + K$ messages for consecutive Q slots, then sends M messages for one slot, and continues to alternate between these two patterns. We fix M as 6, Q as 4, T as 3600s and vary K as 6, 12, 18, 24, 30. We measure the average delivery ratio and delivery delay of normal nodes' messages. The delivery ratio is the percentage of messages that is successfully delivered to the destinations over the total number of messages. The delivery delay is the time taken since the message is created until the time it is delivered to the destination.

Fig. 13 shows the average forwarding priority of attackers compared to normal nodes over time. The forwarding priorities of attackers are degraded over time and become lower than those of the normal nodes. Even if they might not be detected as malicious due to the maximum burst attack, their messages will be given less priorities compared to normal nodes.

Fig. 14 shows the delivery performance of the network with and without FP. When FP is applied, the normal nodes' delivery ratios increase and their delivery delays reduce compared

Fig. 12: Performance of detecting replica count reuse.

Fig. 13: Forwarding priority of normal and malicious nodes over time in FP.

to when FP is not applied. When the number of allowed superfluous messages per time slot K increases in FP, it is observed that the delivery delay increases and the delivery ratio decreases insignificantly. It is noted that the higher K , the more flexibly nodes can send on-demand burst traffic. This means FP helps improving the sending flexibility at the trade-off of slightly degraded delivery performance. Overall, FP shows it works to help ensure the delivery performance of normal nodes even when the adversaries launch the maximum burst attack.

F. Choosing window size

In this section, we investigate the relationship between the window size W and the detection performance of FDER. For the attack of message count reuse, the superfluous messages per time slot is fixed at 3 and the window size is varied from 5 to 40. Fig. 15 shows the detection performances of FDER in this setting. As seen in Fig. 15a, FDER can detect all the malicious nodes accurately even if the window size is set to as small as 5. When the window size increases, the detection delay reduces as in Fig. 15b, i.e. the detection performance is improved. This is because a larger window size contains more information of the malicious nodes' history and hence facilitates FDER to detect faster.

For the attack using replica count reuse, the superfluous replicas per time slot is fixed at 3 and the window size is also varied from 5 to 40. Fig. 16 presents the detection

performances of FDER. From Fig. 16a and 16b, we observe that the optimal window size is 20, since it is the smallest window size that can maintain both a high accuracy and a relatively low detection delay compared to other values.

The overhead of ER history storage and transmission is:

$$\text{Overhead}_{ER} = W \times \text{size}_{ER}$$

where W is the window size of ER history and size_{ER} is the size of each ER. Fig. 17 shows an overview of the fields in the ER and their sizes in bytes. The size of each message record MR is 20 bytes. We run a simulation and obtain the average number of message records in SL (the list of message records sent by a node to another) and RL (list of message records received by a node from another) as 20 for Prophet routing protocol. Thus, the average size of ER is $4 \times 5 + 20 \times 20 + 20 \times$

Fig. 14: Delivery performance of normal nodes with and without FP.

(a) Detection accuracy.

(b) Detection delay.

Fig. 15: FDER's detection of message count reuse attack when varying window size.

Fig. 16: FDER's detection of replica count reuse attack when varying window size.

$20 + 40 \times 2 = 900B$. For the default window size W of 20, the total storage overhead of ER history is $20 \times 900B = 18kB$, which can also be comfortably afforded by mobile devices such as mobile phones and notebooks.

When two nodes meet, they exchange their current ER window, which is at most 18kB as calculated above. For the transmission rate of 3Mbps (that could be achieved in 802.11p standard used in vehicular network [25]), the time taken to transmit the ER overhead is $18kB \div 3Mbps = 48ms$. For the transmission range of 100m and the average velocity of 30km/h in our setting (since the range is 10 to 50km/h), the average contact time between two vehicles is $100m \div (2 * 30km/h) = 6s$. Thus, the transmission time overhead only occupies $48ms \div 6s = 0.8\%$ of the contact time. This is negligible and thus will not affect the available opportunistic duration for message exchange between the nodes.

Fig. 17: ER size.

VI. RELATED WORK

Flooding attack causes Denial of Service (DoS) to legitimate users. In wired network, centralized routers are deployed to monitor the network traffic and malicious flooding messages can be detected and filtered [8], [9]. These schemes cannot be applied to ad-hoc networks which operate without infrastructure support.

MANET is generally assumed as a well-connected network with available end-to-end path for data, making it possible for nodes to know their neighbors' data flow. To prevent flooding attacks in MANET, each node monitors if a neighbor transmits frequently and lowers the forwarding priority of messages from this neighbor if it does [10]–[13]. However, these schemes require flow knowledge, which is hardly available in DTN.

Many pieces of research have been proposed to alleviate flooding attacks in DTN. [14] designed a lightweight authentication to prevent external attackers from accessing the target DTN, but could not detect internal attackers. [15] proposed a queuing mechanism in which each node drops suspiciously flooded messages when its buffer is full. However, it was not general enough to be applied to all DTN routing protocols nor can it eliminate the attacking source. [16], [17] controlled the message generation rate of each node through either centralized server or peer monitoring, but strictly disallowed legitimate users to have burst traffic. Different from these works, FDER is a fully distributed flooding detection and regulation scheme, which is not only independent of the underlying routing protocols but also able to accommodate burst traffic for legitimate users.

VII. CONCLUSION

Being a serious threat to mobile DTN, flooding attack needs to be tackled to ensure network security. Though accommodating burst traffic could enhance user experience in terms of flexibility, it contradicts the security requirement to detect flooding attack since it is hard to differentiate between flooding attack and legitimate burst transmission. To balance these two requirements, we propose a framework comprising FDER - a distributed scheme to detect flooding attack and FP - a fairness policy under the context that nodes are allowed to have short burst transmission. Using ER which is an efficient tool to monitor the sending behaviors of nodes for a long period, FDER could differentiate a persistent flooding attack from a burst transmission over a short time. By integrating

information in ERs window rather than distributing information opportunistically as in Li's scheme, FDER is shown to outperform this state-of-the-art work. The high performance of FDER is achieved at an affordable overhead in terms of storage and transmission. Moreover, FP contributes to alleviating the unintended side effects when enabling burst-traffic transmission, such as unfairness in delivery performance and smart attack which exploits the allowable burst limits. Under such smart attack, FP effectively complements FDER in reducing the impact caused by the adversaries.

REFERENCES

- [1] M. J. Khabbaz, C. M. Assi, and W. F. Fawaz, "Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges", in *IEEE Communications Surveys and Tutorials*, vol. 14, pp. 607-640, 2012.
- [2] Y. Li, P. Hui, D. Jin, and S. Chen, "Delay-tolerant network protocol testing and evaluation", in *IEEE Communications Magazine*, vol. 53, no. 1, pp.258-266, 2015.
- [3] Delay-Tolerant Networking Research Group, <http://www.dtnrg.org/wiki>.
- [4] A. D. J dos Santos, L. H. M. Costa, M. D. L. Braga, P. B. Velloso, and Y. Ghamri-Doudane, "Characterization of a delay and disruption tolerant network in the Amazon basin", in *Vehicular Communications*, vol. 5, pp.35-43, 2016.
- [5] S. Ehsan, K. Bradford, M. Brugger, B. Hamdaoui, Y. Kovchegov, D. Johnson, and M. Louhaichi, "Design and analysis of delay-tolerant sensor networks for monitoring and tracking free-roaming animals", in *IEEE Transactions on Wireless Communications*, vol. 11, no. 3, pp.1220-1227, 2012.
- [6] N. Uchida, N. Kawamura, N. Williams, K. Takahata, and Y. Shibata, "Proposal of delay tolerant network with cognitive wireless network for disaster information network system", in *IEEE 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 249-254, 2013.
- [7] P. R. Pereira, A. Casaca, J. J. Rodrigues, V. N. Soares, J. Triay, and C. Cervell-Pastor, "From delay-tolerant networks to vehicular delay-tolerant networks", in *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp.1166-1182, 2012.
- [8] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks", in *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp.2046-2069, 2013.
- [9] M. H. Bhuyan, H. J. Kashyap, D. K. Bhattacharyya, and J. K. Kalita, "Detecting distributed denial of service attacks: methods, tools and future directions", in *The Computer Journal*, 2013.
- [10] R. H. Jhaveri, S. J. Patel, and D. C. Jinwala, "DoS attacks in mobile ad hoc networks: A survey", in *IEEE 2012 Second International Conference on Advanced Computing & Communication Technologies*, pp. 535-541, 2012.
- [11] S. K. Shandilya, and S. Sahu, "A trust based security scheme for RREQ flooding attack in MANET", in *International journal of computer applications*, vol. 5, no. 12, 2010.
- [12] M. Patel, S. Sharma, and D. Sharan, "Detection and prevention of flooding attack using SVM", in *International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 533-537, 2013.
- [13] Y. Guo, S. Gordon, and S. Perreau, "A flow based detection mechanism against flooding attacks in mobile ad hoc networks", in *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 3105-3110, 2007.
- [14] G. Ansa, H. Criuckshank, Z. Sun, and M. Al-Siyabi, "A DOS-resilient design for delay tolerant networks", in *IEEE International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 424-429, 2011.
- [15] F. C. Lee, W. Goh, and C. K. Yeo, "A queuing mechanism to alleviate flooding attacks in probabilistic delay tolerant networks", in *IEEE 6th Advanced International Conference on Telecommunications (AICT)*, pp. 329-334, 2010.
- [16] V. Natarajan, Y. Yang, and S. Zhu, "Resource-misuse attack detection in delay-tolerant networks", in *IEEE 30th International Performance Computing and Communications Conference (IPCCC)*, pp. 1-8, 2011.
- [17] Q. Li, W. Gao, S. Zhu, and G. Cao, "To lie or to comply: defending against flood attacks in disruption tolerant networks", in *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 3, pp. 168-182, 2013.
- [18] T. N. D. Pham and C. K. Yeo, "Detecting colluding blackhole and greyhole attacks in delay tolerant networks", in *IEEE Transactions on Mobile Computing*, vol. 15, no. 5, pp.1116-1129, 2016.
- [19] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks,"in *SIGMOBILE Mobile Computing and Communication Review*, 2003.
- [20] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks", in *Proceedings of IEEE Infocomm*, Barcelona, Spain, Apr. 2006.
- [21] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks,"in *WDTN 05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 252-259, Philadelphia, USA, Aug. 2005.
- [22] S. Zhao, A. Aggarwal, R. Frost and X. Bai, "A survey of applications of identity-based cryptography in mobile ad-hoc networks", in *IEEE Communications surveys & tutorials*, vol. 14, no.2, 2010.
- [23] L. Law, and B. J. Matt, "Finding invalid signatures in pairing-based batches", in *Cryptography and Coding*, LNCS vol. 4887, pp. 34-53, Springer Berlin Heidelberg, 2007.
- [24] A. Keranen, J.Ott, and T. Karkkainen, "The one simulator for dtn protocol evaluation,"in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Simutools 09*, pp. 55:1-55:10, Rome, Italy, Mar. 2009.
- [25] A. M. Abdelgader, and W. Lenan, "The physical layer of the IEEE 802.11 p WAVE communication standard: the specifications and challenges", in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 2, 2014.