www.ijecs.in
*International Journal Of Engineering And Computer Science ISSN:2319-7242*
*Volume 3 Issue 4 April, 2014 Page No. 5338-5345*

# DEFENDING FLOOD ATTACKS BY LIMITING PACKET REPLICATION ON TIME INTERVAL

V.Senthilvel[1], A.Arjun[2]
[1] v.senthil100@gmail.com, [2] arjuncse02@gmail.com
[1]Research Scholar & Asst. Professor, Department of CSE, Arulmigu Meenakshi Amman College of Engg, Tamil Nadu, India.
[2]M.E-Student, Department of CSE, Arulmigu Meenakshi Amman College of Engg , Tamil Nadu, India.

***Abstract:* Disruption Tolerant Networks (DTNs) consist of node which has high mobility and lack of consistent in connectivity between the nodes. To adopt such a situation two nodes can only exchange data when they move into the transmission range of each other. DTNs employ such contact opportunity for data forwarding. Due to the limitation in bandwidth and buffer space, DTNs are vulnerable to flood attacks. In flood attacks, attackers inject as many packets as possible into the network, or instead of injecting different packets the attacker's forward replicas of the same packet to as many nodes as possible. To defend against flood attacks in DTNs A system is proposed in which each node has a limit over the number of packets that it, as a source node, can send to the network in each time interval. Each node also has a limit over the number of replicas that it can generate for each packet. The two limits are used to mitigate packet flood and replica flood attacks***

IndexTerms—DTN, security, flood attack, detection

## I INTRODUCTION

DISRUPTION Tolerant Networks ( DTNs) [1] consist of mobile nodes carried by human beings [2], [3],vehicles[4], [5], etc. DTNs enable data transfer when mobile nodes are only intermittently connected, making them appropriate for applications where no communication infrastructure is available such as military scenarios and rural areas. Due to lack of consistent connectivity, two nodes can only exchange data when they move into the transmission range of each other (which is called a contact between them). DTNs employ such contact opportunity for data forwarding with "store carry- and-forward"; i.e., when a node receives some packets, it stores these packets in its buffer, carries them around until it contacts another node, and then forwards them. Since the contacts between nodes are opportunistic and the duration of a contact may be short because of mobility, the usable bandwidth which is only available during the opportunistic contacts is a limited resource. Also, mobile nodes may have limited buffer face.

Due to the limitation in bandwidth and buffer space , DTNs are vulnerable to flood attacks. In flood attacks, maliciously or selfishly motivated attackers inject as many packets as possible into the network, or instead of injecting different packets the attacker's forward replicas of the same packet to as many nodes as possible. For convenience, we call the two types of attack packet flood attack and replica flood attack ,respectively. Flooded packets and replicas can waste the precious bandwidth and buffer resources, prevent benign packets from being forwarded and thus degrade the network service provided to good nodes .Moreover, mobile nodes spend much energy on transmitting/receiving flooded packets and replicas which ma y shorten their battery life. Therefore, it is urgent to secure DTNs against flood attacks. Although many schemes have been proposed to defend against flood attacks on the Internet [6] and in wireless sensor networks [7], they assume persistent connectivity and cannot be directly applied to DTNs that have intermittent connectivity. In DTNs, little work has been done on flood attacks, despite the many works on routing [8], [4], data dissemination [9], black hole attack [10], wormhole attack [11], and selfish dropping behaviour [12], [13]. We noted that the packets flooded by outsider attackers (i.e.,the attackers without valid cryptographic with valid signatures. Thus, it is still an open problem is to against flood attacks in DTNs.

Our main contribution is a technique to detect if a node has violated its rate limits. Although it is easy to detect the violation of rate limit on the Internet and in telecommunication networks where the egress router and base station can account each user's traffic, it is challenging in DTNs due to lack of communication infrastructure and consistent connectivity. Since a node moves around and may send data to any contacted node, it is very difficult to count the number of packets or replicas sent out by this node. Our basic idea of detection is claim-carry-and-check. Each node itself counts the number of packets or replicas that it has sent out, and claims the count to other nodes; the receiving nodes carry the claims around when they move, exchange some claims when they contact, and cross-check if the claims are inconsistent. If an attacker floods more packets or replicas than its limit, it has to use the same count in more than one claim according to the pigeonhole principle,[1] and this inconsistency may lead to detection. Based on this idea, we use different cryptographic constructions to detect packet flood and replica flood attacks

Because the contacts in DTNs are opportunistic in nature, our approach provides probabilistic detection. The more traffic an attacker floods, the more likely it will be detected. The detection probability can be flexibly adjusted by system parameters that control the amount of claims exchanged in a contact. We provide a lower and upper bound of detection probability and investigate the problem of parameter selection to maximize detection probability under a certain amount of exchanged claims. The effectiveness and efficiency of our scheme are evaluated with extensive trace-driven simulations.

This paper is structured as follows. Section 2 motivates our work. Section 3 presents our models and basic ideas. Sections 4 and 5 present our scheme. Section 6 presents security and cost analysis. Section 7 presents simulation results. The last two sections present related work and conclusions, respectively.

## II RELATED WORK

Our scheme bears some similarity with previous approaches (e.g., [10]) that detect node clone attacks in sensor networks. Both rely on the identification of some kind of inconsistency to detect the attacker. However, their approaches assumes consistent connectivity between nodes which is unavailable in DTNs. Also, they do not handle the long delays of detection.

A few recent works [10], [21], [12], [11], [13] also address Security issues in DTNs. Li et al. [10] studied the black hole attack in which malicious nodes forge routing metrics to attract packets and drop all received packets. Their approach uses a primitive called encounter ticket to prove the existence of contacts and prevent the forgery of routing metrics, but encounter ticket cannot be used to address flood attacks. Li and Cao [13] also proposed a distributed scheme to mitigate packet drop attacks, which works no matter if the attackers forge routing metrics or not Renet al. [11] studied wormhole attacks in DTNs. Chen and Choon [21] proposed a credit-based approach and Shevade etal. proposed a gaming-based approach [12] to provide in- centives for packet forwarding. Privacy issues have also be addressed[11] . However, these works do not address flood attacks. Other works (e.g., Sprite [13]) deter abuse by correlating the amount of network

Resources that a node can use with the node's contributions to the network in terms of forwarding. This approach has been proposed for mobile ad hoc networks, but it is still not clear how the approach can be applied to DTNs, where nodes are disconnected most of the time. Another recent work [14] proposed a batch authentication protocol for DTNs, which ver ifies multiple packet signatures in an aggregated way to save the computation cost. This work is complmentary to ours, and their protocol can also be used in our scheme to further reduce the computation cost of authentication.

Parallel to our work, Natarajan et al. [13] also proposed a scheme to detect resource misuse in DTNs. In their scheme, the gateway of a DTN monitors the activities of nodes and detects an attack if there is deviation from expected behavior. Different from their work that requires a special gateway for counting, our scheme works in a totally distributed manner and requires no special nodes.

## III MOTIVATION

### *A .The Potential Prevalence of Flood Attacks*

Many nodes may launch flood attacks for malicious or selfish purposes. Malicious nodes, which can be the nodes deliberately deployed by the adversary or subverted by the adversary via mobile phone worms [16], launch attacks to congest the network and waste the resources of other nodes. Selfish nodes may also exploit flood attacks to increase their communication throughput. In DTNs, a single packet usually can only be delivered to the destination with a probability smaller than 1 due to the opportunistic connectivity. If a selfish node floods many replicas of its own packet, it can increase the likelihood of its packet being delivered, since the delivery of any replica means successful delivery of the packet. With packet flood attacks, selfish nodes can also increase their throughput, albeit in a subtler manner. For example, suppose Alice wants to send a packet to Bob. Alice can construct 100 variants of the original packet which only differ in one unimportant padding byte, and send the 100 variants to Bob independently. When Bob receives any one of the 100 variants, he throws away the padding byte and gets the original packet.

### *B. The Effect of Flood Attacks*

To study the effect of flood attacks on DTN routing and motivate our work, we run simulations on the MIT Reality trace [17]

We consider three general routing strategies in DTNs.1) Single-copy routing (e.g., [18], [8]): after forwarding a packet out, a node deletes its own copy of the packet. Thus, each packet only has one copy in the network. 2) Multicopy routing (e.g., [19]): the source node of a packet sprays a certain number of copies of the packet to other nodes and each copy is individually routed using the single-copy strategy. The maximum number of copies that each packet can have is fixed. 3) Propagation routing (e.g., [17], [20], [21]): when a node finds it appropriate (according to the routing algorithm) to forward a packet to another encountered node, it replicates that packet to the encountered node and keeps its own copy. There is no preset limit over the number of copies a packet can have. In our simulations, Sim Bet [8], Spray-and-Focus [19] (three copies allowed for each packet) and Propagation are used as representatives of the three routing strategies, respectively. In Propagation, a node replicates a packet to another encountered node if the latter has more frequent contacts with the destination of the packet.

Two metrics are used, The first metric is packet delivery ratio, which is defined as the fraction of packets delivered to their destinations out of all the unique packets generated. The second metric is the fraction of wasted transmissions (i.e., the transmissions made by good nodes for flooded packets). The higher fraction of wasted transmissions, the more network resources are wasted. We noticed that the effect of packet flood attacks on packet delivery ratio has been studied by Burgess et al. [22] using a different trace [4]. Their simulations show that packet flood attacks significantly reduce the packet delivery ratio of single-copy routing but do not affect propagation routing much. However, they do not study replica flood attacks and the effect of packet flood attacks on wasted transmissions.

In our simulations, a packet flood attacker floods packets destined to random good nodes in each contact until the contact ends or the contacted node's buffer is full. A replica flood attacker replicates the packets it has generated to every encountered node that does not have a copy. Each good node generates thirty packets on the 121st day of the Reality trace, and each attacker does the same in replica flood attacks. Each packet expires in 60 days. The buffer size of each node is 5 MB, bandwidth is 2 Mbps and packet size is 10 KB.

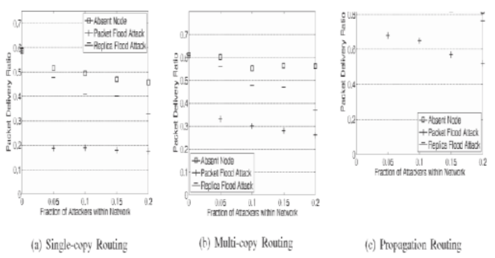


Fig. 1. The effect of flood attacks on packet delivery ratio. In absent node, attackers are simply removed from the network. Attackers are selectively deployed to high-connectivity nodes.

Fig. 1 shows the effect of flood attacks on wasted transmission. Packet flood attack can waste more than 80 percent of the transmissions made by good nodes in all routing strategies when the fraction of attackers is higher than 5 percent. When 20 percent of nodes are attackers, replica flood attack can waste 68 and 44 percent of transmissions in single-copy and multi copy routing, respectively. However, replica flood attack only wastes 17 percent of transmissions in propagation routing. This is because each good packet is also replicated many times.

## IV OVERVIEW

**A. Problem Definition**

**1. Defense against Packet Flood Attacks**

We consider a scenario where each node has a rate limit L on the number of unique packets that it as a source can generate and send into the network within each time interval T.The time intervals start from time 0, T , 2T , etc. The packets generated within the rate limit are deemed legitimate, but the packets generated beyond the limit are deemed flooded by this node. To defend against packet flood attacks, our goal is to detect if a node as a source has generated and sent more unique packets into the network than its rate limit L per time interval.

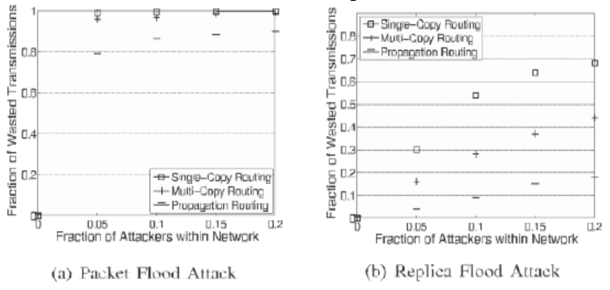


Fig. 2. The effect of flood attacks on the fraction of wasted transmission. Attackers are randomly deployed.

A node's rate limit L does not depend on any specific routing protocol, but it can be determined by a service contract between the node and the network operator as discussed in Section 3.1.3. Different nodes can have different rate limits and their rate limits can be dynamically adjusted.

The length of time interval should be set appropriately. If the interval is too long, rate limiting may not be very effective against packet flood attacks. If the interval is too short, the number of contacts that each node has during one interval may be too nondeterministic and thus it is difficult to set an appropriate rate limit. Generally speaking, the interval should be short under the condition that most nodes can have a significant number of contacts with other nodes within one interval, but the appropriate length depends on the contact patterns between nodes in the specific deployment scenario.

**2. Defense against Replica Flood Attacks**

As motivated in Section 2, the defense against replica flood considers single-copy and multi copy routing protocols. These protocols require that, for each packet that a node buffers no matter if this packet has been generated by the node or forwarded to it, there is a limit l on the number of times that the node can forward this packet to other nodes. The values of l may be different for different buffered packets. Our goal is to detect if a node has violated the routing protocol and forwarded a packet more times than its limit l for the packet.

**3. Setting the Rate Limit L**

One possible method is to set L in a request-approve style. When a user joins the network, she requests for a rate limit from a trusted authority which acts as the network operator. In the request, this user specifies an appropriate value of L based on prediction of her traffic demand. If the trusted authority approves this request, it issues a rate limit certificate to this user, which can be used by the user to prove to other nodes the legitimacy of her rate limit. To prevent users from requesting unreasonably large rate virtual currency (e.g., the credits that she earns by forwarding data for other users [25]) for her rate limit. When a user predicts an increase (decrease) of her demand, she can request for a higher (lower) rate limit. The request and approval of rate limit may be done offline. The flexibility of rate limit leaves legitimate users' usage of the network unhindered. This process can be similar to signing a contract between a smart phone user and a 3G service provider: the user selects a data plan (e.g., 200 MB/month) and pays for it; she can upgrade or downgrade the plan when needed.

**B. Models and Assumptions**

In DTNs, since contact durations may be short, a large data item is usually split into smaller packets (or fragments) to facilitate data transfer. For simplicity, we assume that all packets have the same predefined size. Although in DTNs the allowed delay of packet delivery is usually long, it is still impractical to allow unlimited delays. Thus, we assume that each packet has a lifetime. The packet becomes meaningless after its lifetime ends and will be discarded. We assume that every packet generated by nodes is unique. This can be implemented by including the source node ID and a locally unique sequence number, which is assigned by the source for this packet, in the packet header. We also assume that time is loosely synchronized, such that any two nodes are in the same time slot at any time. Since the inter contact time in DTNs is usually at the scale of minutes or hours, the time slot can be at the scale of one minute. Such loose time synchronization is not hard to achieve.

**C. Basic Idea: Claim-Carry-and-Check**

**1. Packet Flood Detection**

To detect the attackers that violate their rate limit L, we must count the number of unique packets that each node as a source has generated and sent to the network in the current interval. However, since the node may send its packets to any node it contacts at any time and place, no other node can monitor all of its sending activities. To address this challenge, our idea is to let the node itself count the number of unique packets that it, as a source, has sent out, and claim the up-to-date packet count (together with a little auxiliary information such as its ID and a timestamp) in each packet sent out. The node's rate limit certificate is also attached to the packet, such that other nodes receiving the packet can learn its authorized rate limit L. If an attacker is flooding more packets than its rate limit, it has to dishonestly claim a count smaller than the real value in the flooded packet, since the real value is larger than its rate limit and thus a clear indicator of attack. The claimed count must have been used before by the attacker in another claim, which is guaranteed by the pigeonhole principle, and these two claims are inconsistent. The nodes which have received packets from the attacker carry the claims included in those packets when they move around. When two of them contact, they check if there is any inconsistency between their collected claims. The attacker is detected when an inconsistency is found.

**2. Replica Flood Detection**

Claim-carry-and-check can also be used to detect the attacker that forwards a buffered packet more times than its limit l. Specifically, when the source node of a packet or an intermediate hop transmits the packet to its next hop, it claims a transmission count which means the number of times it has transmitted this packet (including the current transmission). Based on if the node is the source or an intermediate node and which routing protocol is used, the next hop can know the node's limit l for the packet, and ensure that the claimed count is within the correct range ½1; l . Thus, if an attacker wants to transmit the packet more than l times, it must claim a false count which has been used before. Similarly as in packet flood attacks, the attacker can be detected.

## V OUR SCHEME

Our scheme uses two different cryptographic constructions to detect packet flood and replica flood attacks independently. When our scheme is deployed to propagation routing protocols, the detection of replica flood attacks is deactivated. The detection of packet flood attacks works independently for each time interval. Without loss of generality, we only consider one time interval when describing our scheme. For convenience, we first describe our scheme assuming that all nodes have the same rate limit L

### *A. Claim Construction*

Two pieces of metadata are added to each packet (see Fig. 4), Packet Count Claim (P-claim) and Transmission Count Claim (T-claim). P-claim and T-claim are used to detect packet flood and replica flood attacks, respectively.

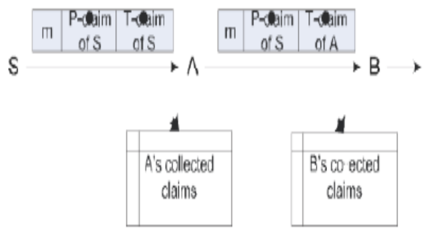


Fig. 4. The conceptual structure of a packet and the changes made at each hop of the forwarding path.

Each hop keeps the P-claim of the source and the T-claim of its previous hop to detect attacks. P-claim is added by the source and transmitted to later hops along with the packet. T-claim is generated and processed hop-by-hop. The source generates a T-claim and appends it to the packet. When an intermediate forwards the packet out, it appends a new T-claim to the packet.

#### *1. P-Claim*

When a source node S sends a new packet m (which has been generated by S and not sent out before) to a contacted node, it generates a P-claim as follows:

P-claim: $S,C_P,T,H(m),SIG_S(H(H(m)|S|CP|T)$

Here, t is the current time. $c_p$ ($c_p$€[1, L]) is the packet count of S, which means that this is the $c_p$th new packet S has created and sent to the network in the current time interval. S increases $c_p$ by one after sending m out. The P-claim is attached to packet m as a header field, and will always be forwarded along with the packet to later hops. When the contacted node receives this packet, it verifies the signature in the P-claim, and checks the value of $c_p$. If $c_p$ is larger than L, it discards this packet, otherwise, it stores this packet and the P-claim.

#### *2. T-Claim*

When node A transmits a packet m to node B, it appends a T-claim to m. The T-claim includes A's current transmission count $c_t$ for m (i.e., the number of times it has transmitted m out) and the current time t. The T-claim is

T-claim: $A,B,H(m),C_T,T,SIG_A,(H(A|B|H(m)|C_T|T))$

B checks if ct is in the correct range based on if A is the source of m. If ct has a valid value, B stores this T-claim. In single-copy and multi copy routing, after forwarding m for enough times, A deletes its own copy of m and will not forward m again.

### *B. Inconsistency Caused by Attack*

In a dishonest P-claim, an attacker uses a smaller packet count than the real value. (We do not consider the case where the attacker uses a larger packet count than the real value, since it makes no sense for the attacker.) However, this packet count must have been used in another P-claim generated earlier. This causes an inconsistency called count reuse, which means the use of the same count in two different P-claims generated by the same node. For example in Fig. 3a the count value 3 is reused in the P-claims of packet m3 and m4. Similarly, count reuse is also caused by dishonest T-claims.

### *C. Protocol*

Suppose two nodes contact and they have a number of packets to forward to each other. Then our protocol is sketched in Algorithm 1.

Algorithm 1. The protocol run by each node in a contact

```
1: Metadata (P-claim and T-claim) exchange and attack detection
2: if Have packets to send then
3: For each new packet, generate a P-claim;
4: For all packets, generate their T-claims and sign them with a hash tree;
5: Send every packet with the P-claim and T-claim attached;
6: end if
7: if Receive a packet then
8: if Signature verification fails or the count value in its P-claim or T-claim is invalid then
9: Discard this packet;
10: end if
11: Check the P-claim against those locally collected and generated in the same time interval to detect inconsistency;
12: Check the T-claim against those locally collected for inconsistency;
13: if Inconsistency is detected then
14: Tag the signer of the P-claim (T-claim, respectively) as an attacker and add it into a blacklist;
15: Disseminate an alarm against the attacker to the network;
16: else
17: Store the new P-claim (T-claim, respectively);
18: end if
19: end if
```

When a node forwards a packet, it attaches a T-claim to the packet. Since many packets may be forwarded in a contact and it is expensive to sign each T-claim separately. The node also attaches a P-claim to the packets that are generated by itself and have not been sent to other nodes before(called new packet in line 3, Algorithm) When a node receives a packet, it gets the P-claim and T-claim included in the packet. It checks them against the claims that it has already collected to detect if there is any inconsistency. Only the P-claims generated in the same time interval (which can be determined by the time tag) are cross-checked. If no inconsistency is detected, this node stores the P-claim and T-claim locally .To better detect flood attacks, the two nodes also exchange a small number of the recently collected P-claims and T-claims and check them for inconsistency.

This meta data exchange process is when a node detects an attacker, it adds the attacker into a blacklist and will not accept packets originated from or forwarded by the attacker. The node also disseminates an alarm against the attacker to other nodes.

### *D. Inconsistency Check*

Suppose node W wants to check a pair of P-claim and T-claim against its local collections to detect if there is any inconsistency. The inconsistency check against full claims is trivial: W simply compares the pair of claims with those collected. In the following, we describe the inconsistency check against compactly stored claims.

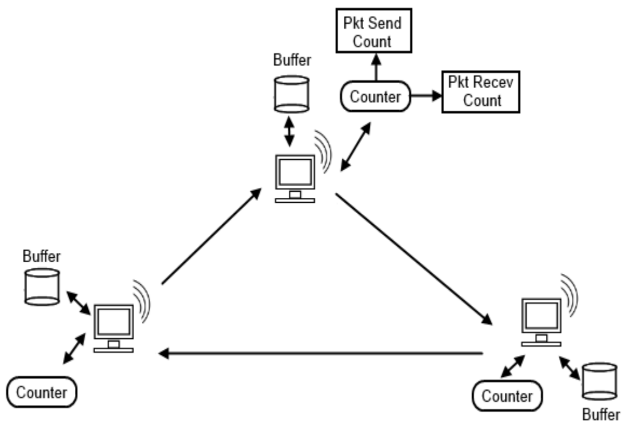
#### *1. Inconsistency Check with P-Claim*

The inconsistency check based on compact P-claims does not cause false positive, since a good node never reuses any count value in different packets generated in the same interval. The inconsistency check may cause false negative if the two inconsistent P-claims have the same hash remainder. However, since the attacker does not know which bits constitute the hash remainder, the probability of false negative is only $2^{8}$. Thus, it has minimal effect on the overall detection probability.

#### *2. Inconsistency Check with T-Claim*

The inconsistency check based on compact T-claims does not cause extra false negative. False positive is possible but it can be kept low as follows: node W may falsely detect a good node R as an attacker if it has received two T-claims generated by R that satisfy two conditions: 1) they are generated for two different packets, and 2) they have the same hash remainder. For 32-bit hash remainder, the probability that each pair of T-claims lead to false detection is 2. In most cases, we expect that the number of T-claims generated by R and received by W is not large due to the opportunistic contacts of DTNs, and thus the probability of false detection is low. As W receives more T-claims generated by R, it can use a longer (e.g., 64-bit) hash remainder for R to keep the probability of false detection low. Moreover, such false detection is limited to W only, since W cannot convince other nodes to accept the detection with compact T-claim.

## VI. SYSTEM ARCHITECTURE



Two pieces of metadata are added to each packet, Packet Count Claim (P-claim) and Transmission Count Claim (T-claim).P-claim and T-claim are used to detect packet flood and replica flood attacks, respectively. Each hop keeps the P-claim of the source and the T-claim of its previous hop to detect attacks. P-claim is added by the source and transmitted to later hops along with the packet. T-claim is generated and processed hop-by-hop. The source generates a T-claim and appends it to the packet. When an intermediate forwards the packet out, it appends a new T-claim to the packet. A node simply compares the pair of claims with those collected. We have inconsistency check for both the P claim and T Claim. To check inconsistency, node first uses node id , packet hash and timestamp to map the P-claim to the structure and reconstructs the hash remainder. The inconsistency check may cause false negative if two inconsistent P-claims have the same hash remainder. In T Claim If receiver id is the node itself then node take no action. They show inconsistency when they are generated for two different packets, and they have the same hash remainder.

## VII METADATA EXCHANGE

When two nodes contact they exchange their collected P-claims and T-claims to detect flood attacks. If all claims are exchanged, the communication cost will be too high. Thus, our scheme uses sampling techniques to keep the communication cost low. To increase the probability of attack detection, one node also stores a small portion of claims exchanged from its contacted node. A node exchanges this small portion of claim to its own future contacts. This is called redirection.

### *A. Sampling*

P-claims and T-claims are sampled together (i.e., when a P-claim is sampled the T-claim of the same packet is also sampled), in the following we only consider P-claims. A node may receive a number of packets (each with a P-claim) in a contact. It randomly samples Z (a system parameter) of the received P-claims, and exchanges the sampled P-claims to the next K (a system parameter) different nodes it will contact, excluding the sources of the P-claims and the previous hop from which these P-claims are received. However, a vulnerability to tailgating attack should be addressed. In tailgating attack, one or more attackers tailgate a good node to create a large number ) of frequent contacts with this node, and send Z packets (not necessarily generated by the attackers) to this node in each created contact. If this good node sends the Z and P-claims of these contacts to the next K good nodes it contacts, much effective bandwidth between these good nodes will be wasted, especially in a large network where K is not small. To address this attack, the node uses an inter-contact sampling technique to determine which P-claims sampled in historical contacts should be exchanged in the current contact. Let SK denote a set of contacts. This set includes the minimum number of most recent contacts between this node and at least K other different nodes. Within this set, all the contacts with the same node are taken as one single contact and a total of Z P-claims are sampled out of these contacts. This technique is not vulnerable to the tailgating attack since the number of claims exchanged in each contact is bounded by a constant.

### *B. Redirection*

There is a stealthy attack to flood attack detection. For replica flood attacks, the condition of detection is that at least two nodes carrying inconsistent T-claims can contact. However, suppose the attacker knows that two nodes A and B never contact. Then, it can send some packets to A, and invalidly replicate these packets to B. In this scenario, this attacker cannot be detected since A and B never contact. Similarly, the stealthy attack is also harmful for some routing protocols like Spray-and-Wait [19] in which each packet is forwarded from the source to a relay and then directly delivered from the relay to the destination. To address the stealthy attack, our idea is to add one level of indirection.

A node redirects the Z P-claims and T-claims sampled in the current contact to one of the next K nodes it will contact, and this contacted node will exchange (but not redirect again) these redirected claims in its own subsequent contacts. Suppose attacker S sends mutually inconsistent packets to two nodes A and B which will never contact. Suppose A and B redirect their sampled P-claims to node C and D, respectively. Then so long as C and B or D and A or C and D can contact, the attack has a chance to be detected. Thus, the successful chance of stealthy attack is significantly reduced.

### *C. The Exchange Process*

Each node maintains two separate sets of P-claims (T-claims, respectively in the following) for metadata exchange, a sampled set which includes the P-claims sampled from the most recent contacts with K different nodes , and a redirected set which includes the P-claims redirected from those contacts. Both sets include Z P-claims obtained in each of those contacts. when two nodes A and B contact, they first select KZ P-claims from each set with the inter-contact sampling technique , and then send these P-claims to each other.

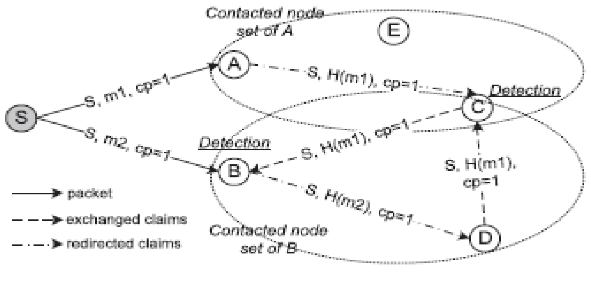


Fig C The idea of redirection which is used to mitigate the stealthy attack

When A receives a P-claim, it checks if this P-claim is inconsistent with any of its collected P-claims using the method If the received P-claim is inconsistent with a locally collected one and the signature of the received P-claim is valid, A detects that the issuer of the received P-claim is an attacker. Out of all the P-claims received from B, A randomly selects Z of the P-claims from the sampled set of B, and stores them to A's redirected set.

### *D. Metadata Deletion*

A node stores the P-claims and T-claims collected from received data packets for a certain time denoted by $\ell$ and deletes them afterward. It deletes the claims redirected from other nodes immediately after it has exchanged them to K different nodes.

### *E. Alarm*

Suppose in a contact a node receives a claim $C_r$ from a forwarded data packet or from the metadata exchange process and it detects inconsistency between $C_r$ and a local claim $C_l$ that the node has collected. $C_r$ is a full claim as shown in Formula 1 (or 2), but $C_l$ may be stored as a full claim or just a compact structure shown in Formula 3 (or 5). If $C_l$ is a full claim, the node can broadcast (via Epidemic routing [19]) a global alarm to all the other nodes to speed up the attacker detection process. The alarm includes the two full claims $C_l$ and $C_r$ . When a node receives an alarm, it verifies the inconsistency between the two included claims and their signatures. If the verification succeeds, it adds the attacker into its blacklist and broadcasts the alarm further; otherwise, it discards the alarm. The node also discards the alarm if it has broadcast another alarm against the same attacker.

If the detecting node stores $C_l$ as a compact structure, it cannot convince other nodes to trust the detection since the compact structure does not have the attacker's signature. Thus it cannot broadcast a global alarm. However, since the attacker may have reused the count value of $C_r$ to other claims besides $C_l$ , the detecting node can disseminate a local alarm that only contains $C_r$ to its contacted nodes who have received those claims. These contacted nodes can verify the inconsistency between $C_r$ and their collected claims, and also detect the attacker. If any of these nodes still stores a full claim inconsistent with $C_r$ , it can broadcast a global alarm as done in the previous case; otherwise, it disseminates anode's local alarm. As this iterative process proceeds, the attacker can be quickly detected by many nodes. Each node only disseminates one local alarm for each detected attacker.

A local alarm and a global alarm against the same attacker may be disseminated in parallel. If a node receives the global alarm first and then receives the local alarm, it discards the local alarm. If it receives the local alarm first, when it receives the global alarm later, it discards the local alarm and keeps the global alarm.

An attacker may falsify an alarm against a good node. However, since it does not have the node's private key (as our assumption), it cannot forge the node's signatures for the claims included in the alarm. Thus, the alarm will be discarded by other nodes and this attack fails.

## VIII ANALYSIS

This section presents rigorous analysis over the security and cost of our scheme, and discusses the optimal parameter to maximize the effectiveness of flood attack detection under a certain amount of exchanged metadata per contact.

### *A. Detection Probability*

The following analysis assumes uniform and independent contacts between nodes, i.e., at any time each node's next contacted node can be any other node with the same probability. This assumption holds for mobility models such as Random Waypoint (RWP) where the contacts between all node pairs can be modeled as i.i.d. Poisson processes [21]. When analyzing the detection probability, we assume that each attacker acts alone

### B. The Basic Attack

First we consider a basic attack in which an attacker S floods two sets of mutually inconsistent packets to two good nodes A and B, respectively. Each flooded packet received by A is inconsistent with one of the flooded packets received by B. In the contacts with A and B, S also forwards some normal, not flooded, packets to A and B to make the attack harder to detect. Let y denote the proportion of flooded packets among those sent by S. For simplicity, we assume y is the same in both contacts. Suppose A and B redirect the claims sampled in the contact with S to C and D, respectively.
To consider the worst case performance, suppose the flooded packets are not forwarded from A and B to other nodes (which is the case in Spray-and-Wait [19]), i.e., only A and B have the inconsistent claims. Note that the analysis also applies to the detection of replica flood attacks.

## IX PERFORMANCE EVALUATIONS

### A. Experiment Setup

To evaluate the performance and cost of our scheme, we run simulations on a synthetic trace generated by the Random Waypoint mobility model and on the MIT Reality trace [17] collected from the real world. In the simulations, 20 percent of nodes are deployed as attackers. They are randomly deployed or selectively deployed to high-connectivity nodes. The buffer size of each node is 5 MB, the Drop Tail policy is used when buffer overflows. The bandwidth is 2 Mbps. Each node generates packets of 10 KB with random destinations at a uniform rete.

### B. Routing Algorithms

**SimBet** [4]: a forwarding-based routing algorithm. It calculates a metric using two social measures (similarity and betweenness), and a packet is forwarded to a node if that node has higher metric than the current one.
**Delegation** [5]: a replication-based routing algorithm, where the receiving node replicates the packet to a neighbor if the neighbor has the highest utility it has seen. We use the contact frequency with the destination as the utility metric.

### C. Metrics

• **Packet delivery ratio**: The percentage of packets delivered to their destinations out of all generated packets.
• **Number of wasted transmissions**: In forwarding-based routing, a transmission is wasted if the transmitted packet is dropped by misbehaving nodes before reaching the destination; in replication-based routing, a transmission is wasted if it directly transmits a packet to a misbehaving node that drops the packet. Here, we only count the wasted packets dropped by routing misbehavior.
• **Detection rate**: The percentage of misreporting nodes detected by normal nodes.
• **Detection delay**: The time needed for misreporting to be deleted
• **Bytes transmitted per contact**: The number of bytes transmitted in a contact for control
• **Bytes stored per node**: The number of bytes stored at a node for control.

### D. Analysis Verification

We use the synthetic trace to verify our analysis results given in Section 6, since in this trace the contacts between node pairs are send to the buffer. [30] which conforms to our assumption for the analysis. We divide the trace into 10 segments, each with 5 $10^4$ time units, and run simulations on each of the third-seventh segments three times with different random seeds. Each data point is averaged over the individual runs.

## X CONCLUSIONS

In this paper, we employed rate limiting to mitigate flood attacks in DTNs, and proposed a scheme which exploits claim-carry-and-check to probabilistically detect the violation of rate limit in DTN environments. Our scheme uses efficient constructions to keep the computation, communication and storage cost low. Also, we analyzed the lower bound and upper bound of detection probability. Extensive trace-driven simulations showed that our scheme is effective to detect flood attacks and it achieves such effectiveness in an efficient way. Our scheme works in a distributed manner, not relying on any online central authority or infrastructure, which well fits the environment of DTNs. Besides, it can tolerate a small number of attackers to collude.
A large part of this paper I have been dedicated to explaining the working procedure, algorithm and proving its efficiency theoretically compare with the existing system models. At the same time, despite the relatively complex proof, the actual algorithm is quite short and easy to implement, as shown in this paper.

## REFERENCES

[1] K. Fall, “A Delay-Tolerant Network Architecture for Challenged Internets,” Proc. ACM SIGCOMM, pp. 27-34, 2003.
[2] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot,“Pocket Switched Networks and Human Mobility in Conference Environments,” Proc. ACM SIGCOMM, 2005.
[3] Motani, V. Srinivasan, and P. Nuggehalli, “PeopleNet: Engineering a Wireless Virtual Social Network,” Proc. MobiCom,pp. 243-257, 2005.
[4] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, “Maxprop:Routing for Vehicle-Based Disruption-Tolerant Networks,” Proc.IEEE INFOCOM, 2006.
[5] S.J.T.U.Grid Computing Center, “Shanghai Taxi Trace Data,”http://wirelesslab.sjtu.edu.cn/, 2012.
[6] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, Internet Denial of Service: Attack and Defense Mechanisms. Prentice Hall, 2005.
[7] C. Karlof and D. Wagner, “Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures,” Proc. IEEE First Int’l Workshop Sensor Network Protocols and Applications, 2003.
[8] E. Daly and M. Haahr, “Social Network Analysis for Routing in Disconnected Delay-Tolerant MANETs,” Proc. MobiHoc, pp. 32-40,2007.
[9] W. Gao, Q. Li, B. Zhao, and G. Cao, “Multicasting in Delay Tolerant Networks: A Social Network Perspective,” Proc. ACM MobiHoc, 2009.
[10] F. Li, A. Srinivasan, and J. Wu, “Thwarting Blackhole Attacks in Distruption-Tolerant Networks

Using Encounter Tickets," Proc. IEEE INFOCOM, 2009.

[11] Y.Ren,M.C.Chuah,J.Yang, and Y. Chen, "Detecting Wormhole Attacks in Delay Tolerant Networks," IEEE Wireless Comm .Magazine, vol. 17, no. 5, pp. 36-42, Oct. 2010.

[12] U.Shevade,H.Song, L. Qiu, and Y. Zhang,"Incentive-Aware Routing in DTNS," Proc. IEEE Int'l Conf. Network Protocols (ICNP '08), 2008.

[13] Q. Li and G. Cao, "Mitigating Routing Misbehavior in Disruption Tolerant Networks," IEEE Trans. Information Forensics and Security, vol. 7, no. 2, pp. 664-675, Apr. 2012.

[14] H. Zhu, X. Lin, R. Lu, X.S. Shen, D. Xing, and Z. Cao, "An Opportunistic Batch Bundle Authentication Scheme for Energy Constrained DTNS," Proc. IEEE INFOCOM, 2010.

[15] B. Raghavan, K. Vishwanath, S. Ramabhadran, K.Yocum, and A. Snoeren, "Cloud Control with Distributed Rate Limiting," Proc. ACM SIGCOMM, 2007.

[16] F-SECURE,"F-Secure Malware Information Pages: Smsworm:- Symbos/Feak," http://www.fsecure. com/v-descs/smsworm symbos feak.shtml,2012.

[17] N. Eagle and A. Pentland, "Reality Mining:Sensing Complex Social Systems," Personal and Ubiquitous Computing, vol. 10, no. 4, pp. 255-268, 2006.

[18] Q. Li, S. Zhu, and G. Cao, "Routing in Socially Selfish Delay Tolerant Networks," Proc. IEEE INFOCOM, 2010.

[19] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "EfficientRouting in Intermittently ConnectedMobile Networks: The Multiple-Copy Case,"IEEE/ACM Trans. Networking, vol. 16, no. 1, pp.77-90, Feb. 2008.

[20] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic Routing in Intermittently Connected Networks," ACM SIGMOBILE Mobile Computing and Comm. Rev., vol. 7, no. 3, pp. 19-20, 2003.

[21] W. Gao and G. Cao, "On Exploiting Transientcontact Patterns for Data Forwarding in Delay Tolerant Networks," Proc. IEEE 18th Int'l Conf. Networks Protocols (ICNP), 2010.