International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163
Volume 1 Issue 9 (October 2014) www.ijirae.com

Claim Algorithm to Avoid Flood Attacks in Disruption Tolerant Networks

[1]P.Akhila, [2] K. Karthik
[1] M.Tech Student, Dept. of CSE, B.V. Raju Institute of Technology, Medak,India
[2] Associate Professor, Dept. of CSE, B.V. Raju Institute of Technology, Medak,India

ABSTRACT—Disruption Tolerant Networks use the moving of nodes and opportunistic contacts among nodes for data communication. Owing to the restriction in network resources such as contact prospect, buffer space and bandwidth. By occurring of flood attacks DTNs are liable to danger. The problem of flood attacks is hear. Attackers send as many packets or packet replicas as possible to the network, in order to misuse the limited network resources. In this paper, we employ rate limiting to protect against flood attacks in DTNs, such that every node has a bound over the number of packets that it can generate in each time interval and a bound over the number of replicas that it can generate for each packet. We propose a distributed scheme to detect if a node has broken its rate limits. To address the challenge that it is difficult to count all the packets or replicas sent by a node due to lack of communication infrastructure, our detection adopts claim-carry-and-check: where every node counts the number of packets or replicas that it has sent and send that count to other nodes, a particular node after receiving the counts from the contacted nodes, just carry that claims when they travel across the network, and cross-check if their carried claims are conflicting when they communicate with other nodes. Hear the claim structure uses the pigeonhole principle to guarantee that an attacker will make inconsistent claims which may lead to detection. Using Rate limit certificate only the flood attacker who exceeds the rate limit was identified.
KEYWORDS— DTN, security, flood attack.

1. INTRODUCTION

A disruption-tolerant network (DTN) is a network intended so that temporary or intermittent communications problems, limitations and anomalies have the least possible adverse impact. Disruption Tolerant Networks (DTNs) [1] has transportable nodes usually carried by human beings [5], [6], vehicles [8], 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. Two nodes can only swap data when they are in the particular communication range of each other because of lack of consistent connectivity. In DTNs data forwarding takes place using one technique called “store-carry-and forward”[12]. This technique works as follows, when anode obtains some packets, it stores these packets in its buffer, carries them until it communicates other node, and then forwards those buffered packets to them. The usable bandwidth available during the contacts is limited resource because the contacts between nodes are opportunistic and the contact may be short duration for the reason that of mobility. In addition to that mobile nodes may have restricted buffer space. Owing to the restriction in bandwidth and buffer space, DTNs are exposed to flood attacks. In flood attacks, cruelly or egoistically stimulated attackers instill as many packets as possible into the network, or instead of inserting different packets the attackers 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. The expensive bandwidth and buffer resources are usually wasted by these flood attacks.

Although many schemes have been proposed to defend against flood attacks on the Internet and in wireless sensor networks [7], they assume persistent connectivity and cannot be directly applied to DTNs that have intermittent connectivity. In DTN Rate limiting was employed to defend against flood attacks in DTNs. In this approach, 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 (i.e., the number of nodes that it can forward each packet to). The two limits are used to mitigate packet flood and replica flood attacks, respectively If a node violates its rate limits, it will be detected and its data traffic will be filtered. In this way, the amount of flooded traffic can be controlled. In DTNs, little work has been done on flood attacks, despite the many works on routing[8],[4] data dissemination[9]black hole attack wormhole attack[11], and selfish dropping behavior[12][13]We noted that the packets flooded by outsider attackers (i.e., the attackers without valid cryptographic credentials) can be easily filtered with authentication techniques.[14] However, authentication alone does not work when insider attackers (i.e., the attackers with valid cryptographic credentials).

Here main objective is to detect node that violates the rate limit and mark rate limits exceeding node as attacker. On the Internet and in telecommunication network it is easy to find out the violation of ratelimit because we have the egress router and base station for accounting each user’s traffic. But it is challenging in DTNs due to lack of communication structure and constant 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. Basic idea of finding inconsistency is claim-carry-and-check. Each node itself calculates 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 across the network, swap some claims when they contact, and cross-check if these

claims are conflicting. If an attacker forwards more packets or replicas than its limit, it has to use the same count in more than one claim according to the pigeonhole principle and this inconsistency may lead to detection. Using this technique, only Attackers who exceed the rate limit can be identified. 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. The effectiveness and efficiency of our scheme are evaluated with extensive trace-driven simulations.

2. ABOUT FLOOD ATTACKS

2.1 Occurrence 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.

2.2 The effect of flood attacks:

To study the effect of flood attacks
We consider three general routing strategies in DTNs.

1) Single copy routing: after forwarding [8]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: 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: when anode 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, Spray-and- Focus (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.

2.3 Collusion Analysis:

2.3.1 Packet Flood Attack

One attacker may send a packet with a dishonest packet count to its colluder, which will forward the packet to the network. Certainly, the colluder will not exchange the dishonest P-claim with its contacted nodes. However, so long as the colluder forwards this packet to a good node, this good node has a chance to detect the dishonest claim as well as the attacker. Thus, the detection probability is not affected by this type of collusion.

2.3.2 Replica Flood Attack

When attackers collude, they can inject invalid replicas of a packet without being detected, but the number of flooded replicas is effectively limited in our scheme. More specifically, in our scheme for a unique packet all the M colluders as a whole can flood a total of M 1 invalid replicas without being detected. To the contrast, when there is no defense, a total of N M invalid replicas can be injected by the colluders for each unique packet. Since the number of colluders is not very large, our scheme can still effectively mitigate the replica flood attack.

3. EXISISTING SCENARIO

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 space. 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. We consider a scenario where each node has a rate limit Lon 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. Hear bandwidth and buffer space decreased andNetwork performance is decreased by DTN nodes performance in this wayDTNs are vulnerable to flood attacks.

4. PROPOSED SCENARIO

To overcome this in the proposed system we employ rate limiting to defend against flood attacks in DTNs. In our approach, 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 (i.e., the number of nodes that it can forward each packet to). The two limits are used to mitigate packet flood and replica flood attacks, respectively. If a node violates its rate limits, it will be detected and its data traffic will be filtered. In this way, the amount of flooded traffic can be controlled. Our main contribution is a technique to detect if a node has violated its rate limits. Our basic idea of detection is claim-"carry-and-check". Hear we use 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.

5. OVERVIEW

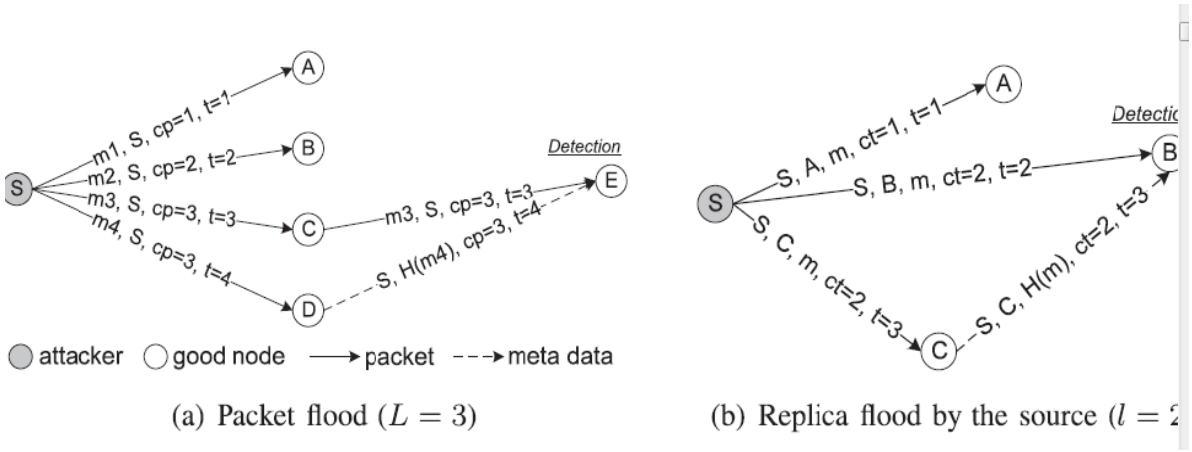
5.1Defense against Packet Flood Attacks:

We consider a scenario where each node has a rate limit Lon 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. 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. 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.

5.2 Defense against Replica Flood Attacks:

As motivated in Section 2, the defense against replica flood considers single-copy and multicopy 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. A node's limit l for a buffered packet is determined by the routing protocol. In Multicopy routing, l ¼ L0(where L0is a parameter of routing) if the node is the source of the packet, and l ¼ 1 if the node is an intermediate hop (i.e., it received the packet from another node). In Singlecopy routing, l ¼ 1 no matter if the node is the source or an intermediate hop. Note that the two limits L and l do not depend on each other. We discuss how to defend against replica flood attacks for quota-based routing

The following diagram shows packet flood and Replica flood attacks:



(a) Packet flood ($L = 3$) (b) Replica flood by the source ($l = 2$

5.3 Claim constriction:
Two pieces of metadata are added to each packet. They are 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. 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. Specifically, the source generates a T-claim and appends it to the packet. When the first hop receives this packet, it peels off the T-claim; when it forwards the packet out, it appends a new T-claim to the packet. This process continues in later hops. Each hop keeps the P-claim of the source and the T-claim of its previous hop to detect attacks.

5.3.1 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,Cp,t,H(m),SIGs(H(H(m)|S|Cp|t))
Here, t is the current time. cp (cp 2 ½1; L_) is the packet count of S, which means that this is the new packet S has created and sent to the network in the current time interval. S increases cp 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 cp. If cp is larger than L, it discards this packet; otherwise, it stores this packet and the P-claim.

5.3.2 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 ct for m (i.e., the number of times it has transmitted m out) and the current time t.

T-claim:
A,B,H(m),Ct,t,SIGa(H(A|B|H(m)|Ct|,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 Multicopy routing, after forwarding m for enough times, A deletes its own copy of m and will not forward m again.

5.3.3 Algorithm:
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

5.4 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 limits, a user pays an appropriate amount of money or virtual currency (e.g., the credits that she earns by forwarding data for other users [17]) 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.

5.5 Trusted Authority:

When a user joins the network, the user 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 user file size. If the trusted authority approves this request, it issues a rate limit certificate to this user, which can be used by the user [17] to prove to other nodes the legitimacy of 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 ratelimit leaves legitimate users‟ usage of the network unhindered. So that the certificate is verified &send to user.

5.6 Claim Detection:

Claim-carry-and-check can also be used to detect the attacker that forwards a buffered packet more times than its limit. 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 for the packet, and ensure that the claimed count is within the correct range. Thus, if an attacker wants to transmit the packet more than its limit, it must claim a false count which has been used before. Similarly in packet flood attacks, the attacker can be detected. 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. Main 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. In the same way replica attackers also identified.

6. CONCLUSION

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.

7. FUTURE WORK

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.

ACKNOWLEDGEMENT: I am really thankful to my guide, Associate professor Mr.K.Karthik for giving such a precious guidelines in completing 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] M. 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] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot,"Pocket Switched Networks and Human Mobility in ConferenceEnvironments," Proc. ACM SIGCOMM, 2005.
[6] M. Motani, V. Srinivasan, and P. Nuggehalli, "PeopleNet:Engineering a Wireless Virtual Social Network," Proc. MobiCom, pp.243-257, 2005.
[7] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, Internet Denial ofService: Attack and Defense Mechanisms. Prentice Hall, 2005.
[8] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, "Maxprop:Routing for Vehicle-Based Disruption-Tolerant Networks," Proc. IEEEINFOCOM, 2006.

[9]W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in DelayTolerant Networks: A Social Network Perspective," Proc. ACMMobiHoc, 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 WormholeAttacks 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-AwareRouting in DTNS," Proc. IEEE Int'l Conf. Network Protocols (ICNP '08), 2008.
[13] Q. Li and G. Cao, "Mitigating Routing Misbehavior in DisruptionTolerant 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, "AnOpportunistic 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.f-secure.com/v-descs/smswormsymbos feak.shtml, 2012.
[17]S.J.T.U.Grid Computing Center, "Shanghai Taxi Trace Data,"http://wirelesslab.sjtu.edu.cn/, 2012.

