International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-4, July 2015
ISSN: 2395-3470
www.ijseas.com

Alleviation of Interest flooding attacks using Token bucket with per interface fairness and Countermeasures in Named Data Networking.

Arani Mantena[1], Rameswara Anand P[2]
[1] Department of Computer Science, Jigjiga Univeristy, Ethiopia.
[2] Department of Computer Science, Jigjiga Univeristy, Ethiopia.

Abstract

Distributed Denial of Service (DDoS) attacks are an ongoing problem in today's Internet. In this paper we focus on DDoS attacks in Named Data Networking (NDN). NDN is a specific candidate for next-generation Internet architecture designs. In NDN locations are named instead of data, So NDN transforms data into a first-class entity and makes itself an attractive and practical approach to meet the needs for many applications. In a Named Data Networking (NDN), end users request data by sending Interest packets, and the network delivers Data packets upon request only, effectively eliminating many existing DDoS attacks. However, an NDN network faces a new type of DDoS attack, namely Interest packet flooding. In this paper we try to alleviate the Interest flooding using token bucket with per interface fairness algorithm and also we try to find countermeasures for Interest flooding attacks (IFA) in NDN.

Keywords: *Future Internet architecture, named-data networking, Distributed denial-of-service, Interest Packet flooding.*

1. Introduction

While the Internet has been a Great success story the emerging usage models and new access methods expose some limitations of the current architecture, that was conceived back in 1970-s. To this end, there have been recent research efforts (e.g., [9], [6], [8], [2]) with the long-term goal of designing and deploying next-generation Internet architecture. One such effort is Named Data Networking (NDN) [7].

NDN is an instance of Content-Centric Networking, where content - rather than hosts - is named and occupies the central role in the communication architecture. NDN is one of five NSF-sponsored Future Internet Architectures (FIA) [3] projects, and is an on-going research effort.

NDN is primarily oriented towards efficient large-scale content distribution. Consumers in NDN directly request (i.e., express *interest* in) pieces of content by name; the network is in charge of finding the closest copy of the content, and of retrieving it as efficiently as possible.
One of the key goals of NDN is *"security by design"*. In contrast to today's Internet, where security problems were (and are still being) identified, the NSF FIA program stresses both awareness of issues and support for features and counter-measures from the outset. To this end, our work investigates distributed denial of service (DDoS) attacks in NDN.

2. NDN overview

NDN supports two types of messages: *interests* and *content.* Interests implement consumer requests and carry a (human-readable) name that identifies the desired content; content messages include a name, a payload and a digital signature computed by the content producer. Names are composed of one or more components, which have a hierarchical structure.

Content is delivered to consumers only upon explicit request. Each request corresponds to an *interest* message and causes NDN routers to store a small amount of transient state in a structure called PIT (Pending Interest Table). This information is used to route content back to consumers.
If the PIT is completely full, new interests are dropped. Hooding a router with interests allows the adversary to saturate the PIT. This has been identified in previous work under the name of interest flooding attack (IFA) [10].

Communication in NDN adheres to the *pull* model: consumer requests content by sending an *interest packet.* If an entity (a router or a host) can "satisfy" a given interest, it returns the corresponding *content packet* Interest and content are the only two types of packets in NDN. A content packet with name X is never forwarded anywhere unless it is requested by an interest for name X.

NDN routers include the following components:
• Content *Store* (CS), used for content caching and retrieval that passes through this router;
• Forwarding *Interest Base* (FIB), that contains a table of name prefixes and corresponding outgoing interfaces;
• Pending *Interest Table* (PIT), a table containing currently unsatisfied interests and corresponding incoming interfaces;

When a router receives an interest for name X and there are no currently pending interests for X in its PIT, it forwards the interest to the next hop, according to its FIB.[1] For each forwarded interest, a router stores some amount of state information, including the name in the interest and the interface on which it arrived.
Otherwise, if an interest for X arrives while there is already an entry for X in the PIT, the router collapses the incoming interest (and any subsequent interests for X), storing only the interface on which it was received. When content is eventually returned- the router looks up its PIT.

[1]The FIB of an NDN router is similar to that of an IP router, except that it associates name prefixes (rather than IP prefixes) to outgoing interfaces. Also, for a given name prefix a router's FIB may contain multiple interfaces; the NDN routers' *strategy module* makes decisions on how to forward interests according to the FIB and additional external information, such as link congestion.
If a PIT entry matching the content name is found, the router forwards the content out on all interfaces appearing in such entry. Additionally, it flushes the corresponding PIT entry. If no matching PIT entry is found, then the content is dropped.
Note that, content always follows, in reverse, the path represented by PIT entries in NDN routers. No other information is needed to deliver content. In particular, an interest does not carry a "source address". Any NDN router can provide content caching via its CS. Thus, content might be fetched from any number of in-network router caches, rather than from its original producer. As a result, unlike IP packets, an NDN interest does not include a "destination address".

3. Interest Flooding attacks in NDN

Interest packets in NDN are routed through the network based on content name prefixes and consume memory resources at intermediate routers.

This makes them a potential tool to launch DDoS attacks in NDN. An attacker or a set of distributed attackers can inject excessive number of Interests in an attempt to overload the network and cause service disruptions for legitimate users (Fig. [1]).
Since an NDN network fetches data by its name, an adversary cannot easily target specific routers or end-hosts. However, an adversary can target a specific namespace. A large volume of such malicious Interests can disrupt service quality in NDN network in two ways: *create network congestion* and *exhaust resources on routers.*

Similar to packets in traditional networks, Interest packets in NDN consume a portion of network capacity. A large number of Interest packets might cause congestion and lead to legitimate packets being dropped in the network. In particular, a coordinated DDoS attack could target one specific namespace and concentrate attack

traffic in certain segments of the network, as routing in NDN is based on name prefixes.

As NDN routers maintain per-packet states for each forwarded Interest (i.e., an entry in its PIT), an excessive amount of malicious Interests can lead to exhaustion of a router's memory, making the router unable to create new PIT entries for incoming Interests and disrupting service for legitimate users.

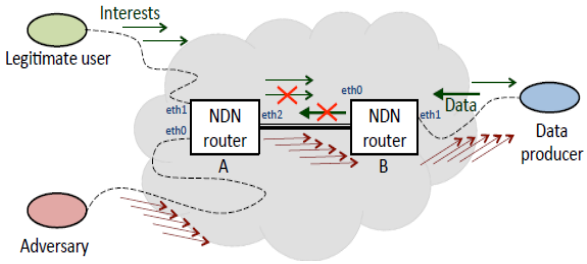

Fig. 1. Example of Interest flooding attack

Nevertheless, creating an effective Interest flooding attack in NDN is non-trivial. To efficaciously target a specific namespace, an adversary needs to make sure that (1) the expressed Interests are routed towards and as close to the data producer/provider as possible, and (2) new corresponding PIT entries are created for those interests and are stored at intermediate NDN routers for as long as possible. The former is achieved when Interests share the same name prefix and as long as they are not served from caches of intermediate routers—an Interest is not forwarded upstream if a router can satisfy it from its content store. The latter requires every single malicious Interest to ask for unique content—all Interests requesting the same content are combined into one PIT entry in routers. Thus, an adversary has to request either an unpopular (i.e. not cached in routers) or non-existing unique content with each Interest. Of the two options available to an adversary, the first one is challenging due to the difficulties around indexing content names in a particular namespace, coordinating a large number of bots to send unique Interests, and sustaining the attack while the network is continuously caching the requested content objects. However, the second option—requesting a unique non-existing content with each Interest—is easy to achieve and sustain. In this paper, we exclusively focus on this particular attack strategy as it not only maximizes the damage from each malicious Interest, but also is the one that is easy to launch and widely applicable to all namespaces (small or large).

4. Interest flooding Alleviation

In this section we present algorithm to alleviate Interest flooding attacks in NDN. Our alleviation strategies feature changing degrees of implementation complexity and effectiveness—if the implementation complexity is high then the algorithm against Interest flooding attacks is more effective. An obvious and simple solution to protect against Interest flooding attacks is to limit the number of Interests forwarded through the network. To this end, we exploit a fundamental principle of NDN architecture—flow balance between Interest and Data packets. Flow balance refers to the fact that one Interest can be satisfied by at most one Data packet. This principle allows intermediate routers to control the inbound data traffic by controlling the number of outstanding Interests in the network. One simple implementation technique is for an NDN router to limit the number of forwarded Interests out of each interface based on the physical capacity of the corresponding interface. This technique is a slight modification of the well-known *Token Bucket* algorithm that is currently widely used in packet-switched networks. Analogous to the *Token Bucket* algorithm, NDN routers can keep track of the amount of data requested that can fully utilize the downstream link (estimated from the number of forwarded Interests) and once the link capacity limit has been reached, they no longer forward new incoming Interests. Ideally, the number of tokens (the pending *Interest Limit)* for each link will be proportional to the link's bandwidth-delay product (BDP) [1]. We can formalize this value as follows:

$$\textbf{Interest Limit} = \textbf{Delay}\ [\textbf{s}].\frac{\textbf{Bandwidth}\ [\frac{\textbf{Bytes}}{\textbf{s}}]}{\textbf{Data packet size}\ [\textbf{Bytes}]} \quad (1)$$

In the above equation, *Delay* is the expected time for the Interest to be satisfied and *Data packet size* is the size of the returning Data packet.

Although both these values are not known a priori, it is not really necessary to use their exact values. One can simply set the pending Interest limit based on the average values of round trip time and observed Data packet size, as network buffers can smooth out most of the network fluctuations.

This *Token Bucket* approach might be exceptionally restrictive in forwarding Interests—not all Interests will result in a Data packet—and might result in underutilization of the network. However, the biggest drawback of this algorithm is the fact that it can nourish DDoS attacks. If a router has utilized all its tokens to forward malicious Interests, it can no longer forward incoming Interests from legitimate users till the pending malicious Interests start to expire. One way to get around this issue is to impose a per interface fairness, so that malicious Interests are not allowed to entirely consume the limits of a specific interface. We describe this technique in greater detail below.

4.1. Token bucket with per interface fairness

To address the lack of fairness associated with the naive *Token Bucket* approach, we modify it to ensure that the Interests forwarded by a router on each interface represent a fair mix of Interests received from neighboring nodes. For example, in Fig.[1] router A can ensure that the tokens associated with Interests sent out on interface eth2 are fairly distributed across incoming interfaces eth0 and ethl. In order to achieve our goal of ensuring "fair" mixing of Interests from all neighboring nodes, we extend the Pending Interest Table to support flagging of Interests that cannot be immediately forwarded and implement hierarchical queues for each interface. This mechanism is essentially a class based queuing [4], with classes for each outgoing and incoming interface. We note that unlike normal queuing. Interest queues do not actually store a packet, but merely a bi-directional pointer to the existing PIT entry. Thus, a PIT entry can be quickly updated when the Interest is actually forwarded, and the element can be easily removed from the queue when the Interest expires.

We present a formalized description of this algorithm in Pseudocode [1]. By setting appropriate queue sizes, we can control the amount of physical resources utilized at a router. It is also important to set a sensible value for how long an Interest can be enqueued. If an Interest is enqueued for a long time, by the time it is dequeued and forwarded, the retrieved Data packet might be dropped at the downstream routers if their corresponding state expired. For our evaluations, we empirically chose to enqueue Interests up to 10% of their original lifetime (100 ms).

Pseudocode 1 Token bucket with per interface fairness

```
1: for each interface if do
2:     L_if ← Interest Limit according to (1)
3:     O_if ← 0                          ▷ Outstanding Interests on interface if

4: function OUTINTEREST(Interest i, InInterface in, OutInterface out)
5:     if L_out − O_out > 0 then                    ▷ out is under limit cap
6:         O_out ← O_out + 1              ▷ "borrow" a token from the bucket
7:         add out to PIT entry and forward i to out
8:     else
9:         Queue q ← out.GetSubQueue(in)
10:        if Size(q) < L_out then
11:            q.PushInterest(i)
12:            add out to PIT entry, and link PIT entry with the queue
13:        else
14:            drop Interest

15:                ▷ Whenever L_out − O_out becomes larger than zero
16: function TOKENBECOMESAVAILABLE
17:     Queue q ← out.GetRoundRobinSubQueue
18:     Interest i ← q.PopInterest
19:     update PIT entry and Forward(i, out)

20: function INDATA(Data d)
21:     lookup PIT entry p for data d
22:     for each outgoing interface out in p do
23:         O_out ← O_out − 1                          ▷ "return" token

24: function TIMEOUT(PIT entry p)
25:     for each outgoing interface out in p do
26:         O_out ← O_out − 1                          ▷ "return" token
```

For this algorithm, we perform 10 independent simulation runs, where we randomly choose 7 client nodes to represent adversaries while the remaining 9 client nodes represent legitimate users. In each run we simulate a 10-minute attack window (total simulation time was 30 minutes, with attack starting at the 10-minute mark). From these simulations, we observe that Token bucket with per interface fairness algorithm provides a

partial relief from Interest flooding attacks, allowing legitimate users to successfully fetch Data for 15-20% of their expressed Interests. We note that while this algorithm might be reasonable for ensuring limited fairness in an NDN network, it is largely ineffective in protecting legitimate users from malicious ones. Attackers are able to successfully thwart access to content for legitimate users by sending a relatively modest volume of malicious Interests.

5. Contribution for countermeasures

Our contribution is two-fold: first, we show that implementing 1FA using limited resources is indeed possible; then we design and evaluate a countermeasure that limits the effect of the attack. Our experiments are based on the official NDN implementation codebases; we argue that this setup provides reliable results, and closely mimics the behavior of physical (non-simulated) networks.

5.1 Effects of IFA

Our simulations show that the most effective way to implement IFA is using interests requesting non-existing content (hereby we use the name *fake* interests to identify them). In fact, state corresponding to such interests is not removed from PITs of intervening routers until it expires. This allows the adversary to quickly and efficiently till up its victim's PIT.

We run simulations on two topologies: a simple architecture and the German research network DFN [5]. We only discuss our results on the (more realistic) DFN topology. We connected two producers, P0 and P1 at opposite sides of the DFN topology. The adversary controls three consumers, which issue only fake interests. While honest users ask for content from P0 and P1, malicious users generate interests that are forwarded only to P0.
Figure 2(b) shows the effectiveness of the attack. In particular, the adversary is able to significantly reduce the bandwidth allocated to content by one of the routers under attack (cf. baseline in Figure 2(a)).

5.2. Countermeasures

There are several parameters that routers can monitor to determine whether they have been targeted by a (successful) IFA. For example, a completely tilled up PIT or a very low bandwidth available to forward content are very good indicators of an in-progress attack. However, routers that are carrying traffic from the attacker may not be able to identify such traffic as malicious.

For this reason, we focus on collaborative detection mechanisms that allow routers to exchange information about their state.
We suggest to periodically check some router's parameters (PIT usage and rate of unsatisfied interests). If they exceed a threshold, the router assumes that there is an IFA in progress. Routers react by rate-limiting interests from attacked interface(s) and by sending "alert" packets, containing information about the attack, to their neighbors. When a router receives an "alert" packet, it lowers the thresholds used to detect IFA. This allows routers that are not direct targets of the attack to detect (and mitigate) IFA.

Figure 2(c) shows the effects of our countermeasure. Due to the limited space, we only report results for one router in the DFN topology (figures 2(a), 2(b), and 2(c)). In particular, this router receives interests (fake and legitimate) directed to the producer under attack. Figure 2(a) shows throughput for content forwarded by the router without IFA. The thick line represents throughput for content coming from P0, while the thin line refers to the throughput of content from P1. Figure 2(b) shows how IFA reduces the bandwidth allocated to content, while Figure 2(c) shows the throughput of the incoming content on routers running our countermeasure. As shown in Figure 3 for different routers in the considered network (on x-axis), our solution is able to provide almost the same throughput as in the baseline scenario (y-axis).

Figure 3. Effects of IFA and of our countermeasure with respect to the baseline.

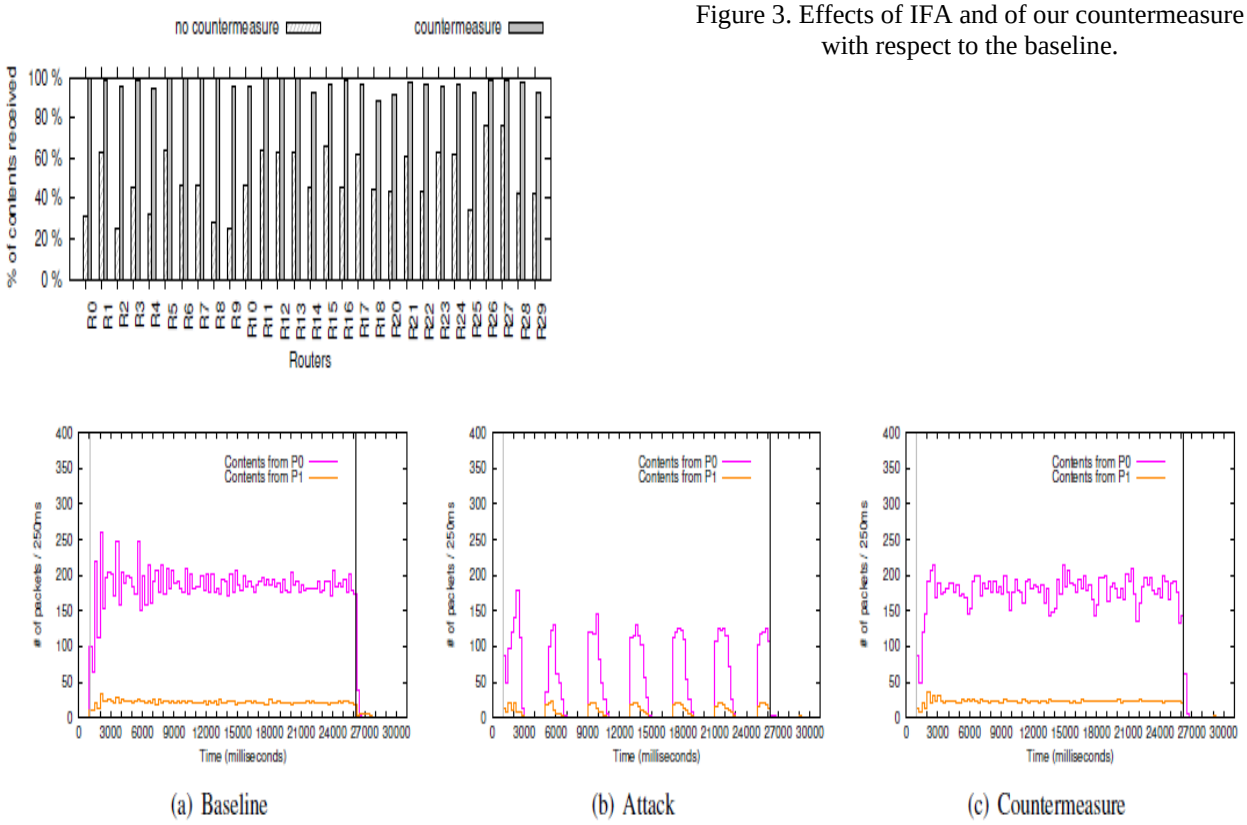

Figure 2. IFA and countermeasure effectiveness compared to baseline. Values shown refer to bandwidth allocated for content by one of the routers forwarding fake interests.

6. Conclusions

This work identifies and characterizes IFA, an effective DDoS attack on NDN. As an initial step in we first examined a specific instance of DDoS attacks—namely Interest flooding—and the severe service degradation such an attack may cause to legitimate users. We performed detailed simulations on a range of topologies to quantify the effectiveness of our algorithm. We demonstrate that IFA is a realistic threat; in particular, we show that an adversary with limited resources can significantly reduce the amount of bandwidth allocated to content. We introduce our techniques to detecting and mitigate IFA. As part of future work, we intend to further improve the effectiveness of our techniques introducing new metrics for early detection of IFA.

References

[1]. A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock , "Host-to-host congestion control for TCP," *IEEE Comm. Surveys and Tutorials.* vol. 12, no. 3, July 2010.

[2]. ChoiceNet - Evolution through Choice. https://code.renci.org/ gf/project/choicenet/.

[3] National science foundation (NSF) future of internet architecture (FIA) program, http://www.nets-ria.net/.

[4] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365-386, 1995.

[5]O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz. On realistic network topologies for simulation. In *ACM SIGCOMM MoMeTools.* pages 28-32. ACM Press, 2003.

[6] MobilityFirst FIA Overview. http://mobilityfirst.winlab.rutgers.edu.

[7] Named data networking project **(NDN).** http://named-data.org.

[8] Nebula, http://nebula.cis.upenn.edu.

[9] XIA - eXpressive Internet Architecture, http://www.cs.cmu. edu/~xia/.

[10] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. DoS & DDoS in named-data networking. Technical report, University of California. Irvine, 2012.

[11] Alexander Afanasyev, Priya Mahadevan, Ilya Moiseenko, Hrsin Uzun, Lixia Zhang, " Interest Flooding Attack and Countermeasures Named Data Networking". IFIP networking conference 2013.