Interest Flooding Attack and Countermeasures in Named Data Networking

Alexander Afanasyev*, Priya Mahadevan†, Ilya Moiseenko*, Ersin Uzun†, Lixia Zhang*
*University of California, Los Angeles
{afanasev, iliamo, lixia}@cs.ucla.edu
†Palo Alto Research Center
{ersin.uzun, priya.mahadevan}@parc.com

***Abstract*—Distributed Denial of Service (DDoS) attacks are an ongoing problem in today's Internet, where packets from a large number of compromised hosts thwart the paths to the victim site and/or overload the victim machines. In a newly proposed future Internet architecture, Named Data Networking (NDN), end users request desired data by sending Interest packets, and the network delivers Data packets upon request only, effectively eliminating many existing DDoS attacks. However, an NDN network can be subject to a new type of DDoS attack, namely Interest packet flooding. In this paper we investigate effective solutions to mitigate Interest flooding. We show that NDN's inherent properties of storing per packet state on each router and maintaining flow balance (i.e., one Interest packet retrieves at most one Data packet) provides the basis for effective DDoS mitigation algorithms. Our evaluation through simulations shows that the solution can quickly and effectively respond and mitigate Interest flooding.**

***Index Terms*—Information-centric networks, named-data networking, denial-of-service**

I. Introduction

Named Data Networking (NDN) [1], [2] is an ongoing research effort that aims to move the Internet into the future with a content-centric design that is capable of efficient content distribution and seamless mobility support. In contrast to today's Internet, a key goal of the NDN project is "security by design." In fact, it goes a long way by guaranteeing the integrity and provenance of every Data packet with digital signatures and protecting user-privacy with no source addresses carried in the packets. However, one big question that is yet to be answered is: how does the NDN architecture fare in terms of its resilience against DDoS attacks? Especially since various forms of DDoS attacks pose a significant threat to the existing Internet infrastructure [3], it is crucial to ensure that the new design is free of similar vulnerabilities.

NDN eliminates host-based addressing and makes data the first-class network entity. Instead of sending packets to a given IP address, NDN nodes request desired data by sending Interest packets carrying application-level data names, and the network returns the requested Data packets following the path of Interests. Such a shift automatically eliminates several long-standing DDoS attacks, including direct flooding and reflector attacks through source address spoofing [4]. However, malicious users can attack the network by sending an excessive number of Interests. Since each Interest consumes resources at intermediate routers as it is routed through the network, an excessive number of Interests can congest the network and exhaust a router's memory. We coin the term *Interest flooding* to refer to such attack and this paper exclusively investigates the problem and the solution space for it.

Our effort is an important first step towards a complete investigation of DDoS attacks in NDN. We experiment with three algorithms that allow routers to exploit their state information to thwart these attacks. Through extensive simulations, we show how one of our mitigation methodologies effectively shuts down malicious users while preventing legitimate users from service degradation. The rest of the paper is organized as follows. We provide an overview of NDN architecture in Section II and describe Interest flooding attacks in Section III. In Sections IV and V we introduce techniques to mitigate these attacks, evaluate their effectiveness, and discuss their limitations. We summarize related work in Section VI. We discuss future work and conclude in Section VII.

II. NDN Overview

In this section we briefly introduce NDN with a focus on its stateful forwarding plane (refer to [1], [2], [5], [6] for more details). NDN is a receiver-driven, data-centric communication protocol. All communications in NDN are performed using two distinct types of packets: *Interest* and *Data*. Both types of packets carry a *name*, which uniquely identifies a piece of content that can be carried in one Data packet. Data names in NDN are hierarchically structured and an example name for the first segment of a youtube video would look like: "`/youtube/videos/0F8YdlkKO9A/0`".

To retrieve data, a consumer requests it by sending an Interest packet with the name of the desired content in it. Routers use this name to route the Interest towards data sources, and a Data packet whose name matches the name in the Interest is returned to the consumer by following the reverse path of the Interest. Similar to IP, Interest forwarding is based on longest name prefix match, but, unlike IP, an Interest packet and its matching Data packet always take symmetric paths.

Each NDN router maintains three major data structures:

- *Pending Interest Table (PIT)* holds all "not yet satisfied" Interests that have been sent upstream towards potential data sources. Each PIT entry contains one or multiple incoming and outgoing physical interfaces; multiple

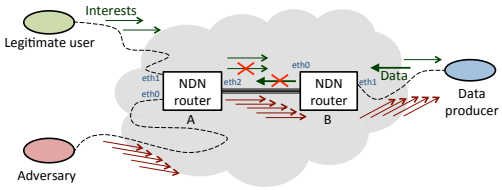

Fig. 1. Example of Interest flooding attack

incoming interfaces indicate the same data is requested from multiple downstream users; multiple outgoing interfaces indicate the same Interest is forwarded along multiple paths.

- *Forwarding Interest Base (FIB)* maps name prefixes to one or multiple physical network interfaces, specifying directions where Interests can be forwarded.
- *Content Store (CS)* temporarily buffers Data packets that pass through this router, allowing efficient data retrieval by different consumers.

When a router receives an Interest packet, it first checks whether there is a matching data in its CS. If a match is found, the Data packet is sent back to the incoming interface of the Interest packet. If not, the Interest name is checked against the entries in the PIT. If the name already exists in the PIT, then it can be a duplicate Interest (identified by a random number each Interest carries) that should be dropped, or an Interest from another consumer asking for the same Data, which requires the incoming interface of this Interest to be added to the existing PIT entry. If the name does not exist in the PIT, the Interest is added into the PIT and forwarded along the interface chosen by the strategy module, which uses FIB as input for its decisions.

When a Data packet is received, its name is used to look up the PIT. If a matching PIT entry is found, the router sends the Data packet to the interface(s) from which the Interest was received, caches the data in the CS, and removes the PIT entry. Otherwise, the Data packet is deemed unsolicited and is discarded. Each Interest also has an associated lifetime; the PIT entry is removed when the lifetime expires. Although the maximum lifetime is specified by users, it is ultimately a router's decision on how long it is willing to keep a PIT entry. For simplicity, we assume routers keep Interests in PIT for one second in our simulations.

III. Interest Flooding Attacks in NDN

As we explained earlier, 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. For example in Fig. 1, if the data producer is the exclusive owner of "`/foo/bar`" namespace, both router B and the data producer would receive all Interests for "`/foo/bar/...`" that cannot be otherwise satisfied from in-network caches.[1] 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.

Nevertheless, creating an effective Interest flooding attack in NDN is non-trivial. To efficaciously target a specific namespace (e.g., "`/newyorktimes/`"), 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 (e.g., "`/newyorktimes/`") 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. For example, an adversary can construct such Interests by concatenating a variable-length random name component to the victim namespace (e.g., "`/newyorktimes/3rf3...`"). 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).

In the rest of this paper, we use the general term *Interest*

[1]This example assumes that the adversary floods the network with unique data names carrying "/foo/bar" prefix to make them effective. It also assumes the producer is single-homed and the data is not replicated elsewhere. With multi-homed producers or replicated data, NDN would likely to cope better with DDoS attacks due its native multipath and adaptive forwarding [5], [6] support.

flooding attack to refer to the above described attack and assume an attacker is limited to controlling a botnet of end-hosts only, i.e., we assume the routers in the network and the computers in the victim domain are not compromised.

IV. Interest flooding mitigation methods

In this section we present several algorithms to mitigate Interest flooding attacks in NDN. Our mitigation strategies feature varying degrees of implementation complexity and effectiveness—the higher the implementation complexity, the more effective is the algorithm against Interest flooding attacks. We start by describing our simple strategies and use the insights and lessons learned from the deployment of these to inform and design more effective mitigation techniques that work well in various topologies.

An obvious and naïve solution to defend against Interest flooding attacks is to restrict 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) [7]. We can formalize this value as follows:

$$\text{Interest Limit} = \mathit{Delay}\ [s] \cdot \frac{\text{Bandwidth [Bytes/s]}}{\text{Data packet size [Bytes]}} \qquad (1)$$

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.

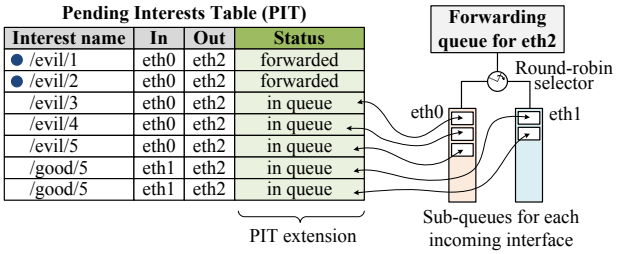

Fig. 2. Interest queuing: if tokens are unavailable, the router creates a PIT entry, but instead of forwarding, it enqueues the Interest

*1) **Token bucket with per interface fairness**:* To address the lack of fairness associated with the naïve *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 `eth1`. 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 (see Fig. 2). This mechanism is essentially a class based queuing [8], 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).

As we show in Section V, this 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.

The key drawback of the *Token bucket with per interface fairness* algorithm is that it still admits a relatively large number of Interests from malicious users. A considerable

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
```

Pseudocode 2 Interest satisfaction statistics

```
1: for each interface if do
2:     F_if ← 0                    ▷ forwarded Interests from interface if
3:     F̂_if ← 0                                  ▷ averaged value of F_if
4:     U_if ← 0                  ▷ unsatisfied Interests from interface if
5:     Û_if ← 0                                  ▷ averaged value of U_if
6: function OUTINTEREST(Interest i, InInterface in)
7:     F_in ← F_in + 1
8:     record in in the list of incoming interfaces for i
9: function INTERESTTIMEOUT(Interest i)
10:    lookup the list of incoming interfaces for i
11:    for each interface if in the list do
12:        U_if ← U_if + 1
13:            ▷ Exponentially weighted moving average smoothing
14: function EWMA                                        ▷ Every second
15:    α ← e^{−1.0/30.0}
16:    for each interface if do
17:        Û_if ← α · Û_if + (1 − α) · U_if
18:        U_if ← 0
19:        if F_if > 0 then          ▷ To ensure decaying of ratio U_if/F_if
20:            F̂_if ← α · F̂_if + (1 − α) · I_if
21:            F_if ← 0                                  ▷ Reset counters
```

percentage of these malicious Interests are forwarded all the way to content producers, thereby reducing resources available to serve legitimate users. This algorithm attempts to ensure that each interface does not forward more than its fair share of Interests, but in doing so, it drops both legitimate and malicious Interests. For any strategy to be effective in defending against Interest flooding attacks, it must be able to detect and differentiate to some extent malicious requests from legitimate ones. Thus, the key question is how can we devise mitigation algorithms that allow a router to distinguish between 'good' and 'bad' Interests?

A. Intelligent attack mitigation

In order to distinguish between legitimate and malicious Interests, we leverage another unique feature of NDN architecture—guaranteed symmetric flow of Interest and Data packets. Since a Data packet takes the reverse path of the corresponding Interest packet, a router is guaranteed to see if an Interest it forwarded resulted in a matching Data packet or timed out. Since malicious Interests are not likely to bring data back (as discussed in Section III), this information can be utilized by routers in differentiating attack and legitimate traffic.

This timeout-based differentiation method is reactive in nature: one cannot determine in advance if an Interest will result in a timeout or Data being retrieved. However, routers can proactively maintain up-to-date statistics of Interest satisfaction ratios (number of forwarded versus number of satisfied Interests), and use these statistics to determine whether an incoming Interest should be forwarded or dropped. For example, maintaining Interest satisfaction ratio statistics for each incoming interface is sufficient to reasonably predict whether an Interest received from a neighbor connected to this interface will result in a Data packet or a timeout if forwarded. Statistics can also be kept at finer granularities such as per outgoing interface, per name prefix, etc. that can further improve the estimates. A router's goal should be to prioritize Interests that bring Data back while quickly penalizing those that occupy resources but do not result in a returning Data packet. In order to allow negative statistics to build up fast and positive statistics to deteriorate quickly, we use the standard exponentially weighted moving average, performed once a second with α coefficient $e^{-1/30}$, approximately corresponding to a 30-second averaging window.

Pseudocode 2 formally defines how statistics can be generated for each incoming interface. Note that in order to ensure decaying of relative statistics (e.g., ratio between the number of unsatisfied and forwarded Interests), only unsatisfied statistics needs to be exponentially smoothed (lines 19–21).

Fig. 3 illustrates the resulting dynamics of the statistics during and after an Interest flooding attack. The attack duration is from 10 to 70 seconds. Prior to start of the attack, the percentage of unsatisfied Interests is zero. The statistics build up rapidly as soon as Interests start to time out, which happens approximately one second after the start of the attack. For the duration of the attack (10–70 seconds), the percentage of unsatisfied Interests is close to 100%: when the ratio is close to 100%, routers drop all incoming Interests, resulting in decaying of the statistics until a new Interest is admitted, which eventually brings statistics back near 100% point. Finally, the ratio exponentially decays after the attack ceases.

*1) **Satisfaction-based Interest acceptance**:* Having successfully implemented a technique to gather statistics on Interest satisfaction ratios, our next challenge is in using these ratios to penalize malicious Interests. A straightforward method to achieve this enforcement is to use the Interest satisfaction ratio as a direct probability for accepting (forwarding) or rejecting

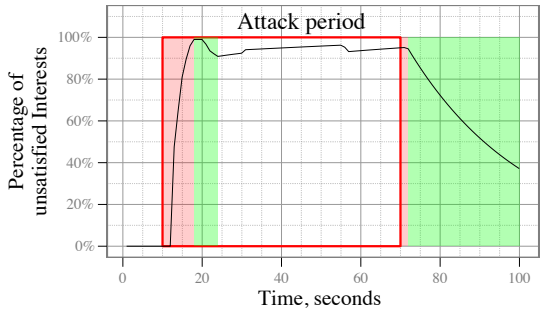

Fig. 3. Dynamics of unsatisfied Interests statistics on a gateway's interface towards the attacker

Pseudocode 3 Satisfaction-based Interest acceptance

```
1:        ▷ Same init, InData and Timeout functions as in Pseudocode 1
2: function OutInterest(Interest i, InInterface in, OutInterface out)
3:                    ▷ Use uniform probability distribution model P(X)
4:                    ▷ P(X) : ∀x ∈ [0,1] ⇒ P(x) = x
5:     if F_in > θ then    ▷ At least some Interests were forwarded before
6:         s ← (1 − U_in/F_in)
7:         Drop interest with probability P(s)
8:     forward the Interest, subjecting to token bucket limits
```

an incoming Interest (see Pseudocode 3).

Parameter θ on line 5 of Pseudocode 3 ensures that the probabilistic model is not enforced when the volume of Interests arriving at a particular interface is small. This step is critical to provide an opportunity for legitimate users to regain their share of resources after temporary Data delivery failures.

A drawback of the satisfaction-based Interest acceptance method is that each router on the path makes an independent decision on whether to forward or drop an Interest. As a result of these independent decisions, the probability of legitimate Interests being forwarded decreases rapidly as the number of hops between the content requester and producer grows; worsening the Interest satisfaction statistics and resulting in further drops. In our example in Fig. 1, the router A observes 50% satisfaction rate for `eth1` and 0% rate for `eth0`. At the same time, router B observes a 30% satisfaction rate for its `eth0` interface. Next time a legitimate Interest arrives at router A, it has a 50% chance of being forwarded further, and if forwarded, it has only a $50\% \times 30\% = 15\%$ probability of being forwarded further towards the data producer. With each increasing hop in the network, the probability of being forwarded to the next hop decreases significantly. One way to prevent this overreaction and unfair penalization is to ensure that the decision taken at each router on whether to forward or drop the Interest is not independent of the decision taken at preceding routers. An explicit notification such as a gossip protocol between neighboring NDN routers might alleviate the problem, but we leave the design and evaluation of it to future work.

*2) **Satisfaction-based pushback**:* The previous algorithm—the satisfaction-based Interest acceptance—divides the available forwarding tokens among all interfaces in proportion to their Interest satisfaction ratios. An alternate algorithm

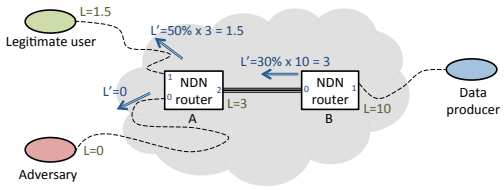

Fig. 4. Satisfaction-based pushback example

Pseudocode 4 Satisfaction-based pushback

```
1:        ▷ Same init, InData and Timeout functions as in Pseudocode 1
2: ∀f ∈ interfaces : L'_f ← L_f      ▷ Per-incoming interface Interest limit
3:                                   ▷ Announcement from the neighbor
4: function InLimits(InInterface in, Limit L')
5:     L_in ← L'
6: function AnnounceLimits                          ▷ E.g., every second
7:     for each outgoing interface out do
8:         for each incoming interface in do
9:             L'_in = L_out × (1 − U_in/F_in)
10:            AnnounceLimit(in, L'_in)
```

for proportional token distribution without overreaction is to enable and enforce explicit Interest limit for each incoming interface, where the value of the limit depends directly on the interface's Interest satisfaction ratio. Routers need to announce these limits to their downstream neighbors, ensuring that any Interest forwarded from the downstream router is allowed to get through, resulting in genuine Interest satisfaction statistics.

The formal definition of the satisfaction-based pushback algorithm is presented in Pseudocode 4, while Fig. 4 illustrates how the algorithm will work in our example in Fig. 1. Assuming an initial token bucket limit $L = 10$ and the current satisfaction ratio for router A is 50% for `eth1` and 0% for `eth0`, and for router B the ratio is 30% for `eth0`, each node will set and announce the following incoming interface limit L':

1) router B will set and announce the incoming interface limit $L' = 3$;
2) router A, after receiving announcement from B will readjust its incoming interface limits to $L'_{eth1} = 1.5$ and $L'_{eth0} = 0$; and
3) both legitimate users and adversaries may either obey or ignore the announced limit, which will in any case be enforced by router A.

The zero limit for the adversary's link implies that router A is temporarily not willing to accept any Interests from this interface until the statistics decay to an appropriate level (recall Fig. 3). At the next iteration of the satisfaction-based pushback algorithm, a legitimate user will be able to gradually improve the statistics on both routers A and B as all Interests from the user will get through and return Data, eventually resulting in a full allowance ($L' = L = 10$) in the links between the routers A and B, and the user and router A.

We note that while in the description of the satisfaction-based pushback algorithm we explicitly used "outgoing" and "incoming" interfaces, all interfaces can be both incoming and

outgoing. Thus, it may not be entirely clear which outgoing limit L_{out} (line 9 in the algorithm) should be used to calculate the incoming limit L_{in}. To overcome this problem, in our actual implementation we enforced separate incoming/outgoing interface limits for each individual FIB entry. That is, for each FIB entry we set a separate Interest limit for each incoming interface (L'^{fib}_{in}) based on the sum of FIB entry limits for each outgoing interface $L = \sum L^{fib}_{out}$.

Both satisfaction-based Interest acceptance and satisfaction-based pushback algorithms are forms of a well-known pushback mechanism [9], but with several core differences. First, we are suppressing (pushing back) unwanted requests for data, not actual data itself. Second, differentiating between good and bad Interests is based on the traffic symmetry principle of NDN. Finally, both intelligent attack mitigation algorithms can be deployed at all times without degrading network performance even when there are no active attackers.

V. Evaluation of Interest flooding mitigation methods

In this section, we present an in-depth evaluation study, aiming to quantify the effectiveness of all our Interest flooding attack mitigation methods. We used the open-source ndnSIM [10] package, which implements NDN protocol stack for NS-3 network simulator (http://www.nsnam.org/), to run simulations for a variety of network topologies and scenarios. We extended ndnSIM with our three mitigation algorithms—token bucket with per interface fairness, satisfaction-based Interest acceptance, and satisfaction-based pushback—and evaluated the effectiveness of each algorithm independently.

The metric we choose to quantify the effectiveness of our algorithms is the *percentage of satisfied Interests for legitimate users*. This metric corresponds to the quality of service experienced by legitimate users when the network is under attack. In other words, if the network implements a mitigation method X and a high percentage of user-expressed Interests are satisfied even while the network is under attack, then one can conclude that method X is highly effective at mitigating the attack.

In our experiments, we assumed that legitimate users express Interests at constant average rates with randomized time gap between two consecutive Interests, where the random number for the gap follows a uniform distribution. We believe that this traffic pattern provides a reasonable approximation of traffic mix from all network users without excessive buffering. To quantify the behavior of our mitigation strategies under a worst-case attack scenario, we assumed that all the attackers send junk Interests as fast as they can. Further, no Interest—including legitimate ones—can be satisfied from caches. We also configured routers for single-path Interest forwarding and there is only one single-homed producer for the prefix under attack.

We ran our simulations on two different network topologies—a smaller binary tree topology and a much larger ISP-like topology. We use a binary tree topology as it represents one of the worst cases to defend against Interest flooding DDoS attacks. The larger ISP topology reflects how our mitigation methods would perform when deployed on the real Internet. Again, to study the performance of our mitigation strategies under a range of conditions, we varied the percentage of attackers in the network—the values ranged from 6% attackers to over 50% attackers in the network. We set the *delay* and *data size* parameters for the Interest limit calculation (formula 1) to a fixed value for every node in the simulated topology. In particular, for the small-scale binary tree topology, we set delay to 80 ms, while for the large-scale ISP topology we set it to 330 ms (the order of the largest RTT). The data size is 1100 bytes for all simulation runs and topologies.

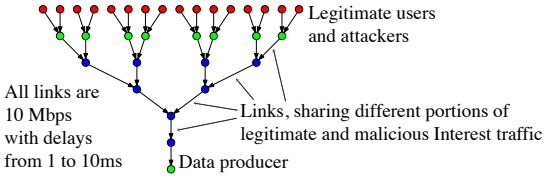

Fig. 5. Small-scale binary tree topology

A. Small-scale evaluations

In Fig. 5, we depict the binary-tree topology that we used for our initial experiments. Legitimate users as well as attackers were placed on leaf nodes (top row of red nodes) as shown in the figure. There are 16 end users (both legitimate and attackers) in this topology, each expressing Interests that are routed towards a single data producer, placed at the root of the tree. Each link in this topology is assigned a bandwidth of 10 Mbps and a randomized propagation delay ranging from 1 to 10 ms.

1) Effectiveness of the three mitigation algorithms: Our goal is to compare the effectiveness of each mitigation method and quantify the percentage of Interests satisfied for all legitimate users while the network is under attack. For each mitigation 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). We plot the minimum and maximum range for observed Interest satisfaction percentages for all legitimate users aggregated across the 10 simulation runs as a function of time for each mitigation algorithm in Fig. 6. Token bucket with per-interface fair queuing performs the worst, while satisfaction-based pushback performs the best, with almost a 100% satisfied Interests for all legitimate users.

*a) **Token bucket with per interface fairness**:* We observe a successful DDoS attack, where 40% attackers succeed in significantly shutting down the remaining 60% legitimate users—a mere 15% of their Interests are satisfied by Data from the producer. Contrary to expectations, the 60% legitimate users do not receive at least 60% of network resources. As described in the previous section, the key limitation of this

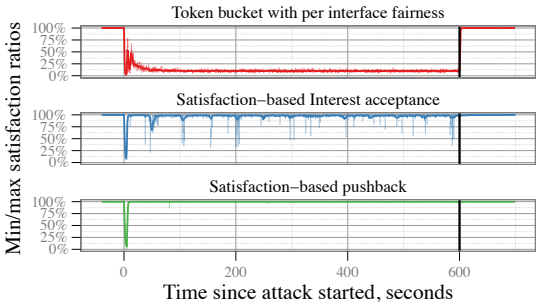

Fig. 6. Interest satisfaction ratio as a function of time for binary-tree topology with 7 attackers and 9 legitimate users

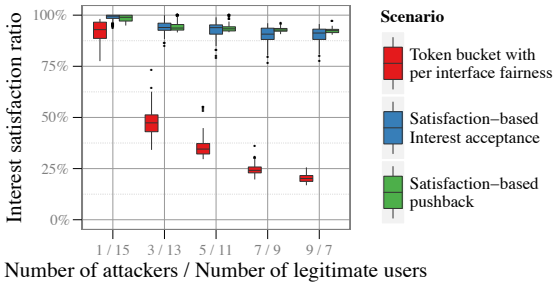

Fig. 7. Average Interest satisfaction ratios for the first minute of the experiment as a function of increasing attackers in the network

algorithm is that it still admits Interests from attackers and causes congestion and packet drops near the victim.

*b) **Satisfaction-based Interest acceptance**:* The effectiveness of this algorithm stems from the fact that routers can differentiate and limit malicious Interests into the network. The observed periodic dips in the Interest satisfaction ratios of legitimate users in Fig. 6 is a direct result of Interest satisfaction rate statistics decaying with time. The 50-second period approximately corresponds to the selected exponential decaying parameter $\alpha = e^{1.0/30.0}$, which decays statistics to $1/e$ of the initial value within 30 seconds and to ≈20% within 50 seconds. When Interests from attackers start to get readmitted, they cause degradation of statistics on routers close to the producer (i.e., routers that observe mixed traffic from legitimate and malicious users). Consequently, this degradation reduces the probability of legitimate Interests getting through (see Section IV-A1) until malicious Interests are "pushed back" again to the edge.

*c) **Satisfaction-based pushback**:* In our simulations, this mitigation algorithm was able to effectively shut down attackers and ensure that almost all the Interests from legitimate users are satisfied. We observe a sharp dip in the satisfaction ratio curve at the start of the attack as it takes a few seconds for all routers to be fully aware of the attack. However, recovery is quick as malicious Interests start to time out and explicit Interest limit announcements start to succeed in containing malicious Interests close to the attacker. Till then, the network, for a short period of time (under 10 seconds for all simulation runs), fails to provide 100% service for legitimate users. Once the malicious Interests are effectively shut down, all Interests from legitimate users are satisfied. Unlike the satisfaction-based Interest acceptance scenario, we do not observe any periodic dips in the satisfaction curve, as the pushback algorithm effectively guarantees that once an Interest is admitted, it will likely be routed all the way to the data producer.

2) Network reaction to varying number of attackers: Our next goal is to study the effectiveness of our mitigation algorithms as a function of increasing adversaries in the network. To this end, we vary the percentage of attackers in the topology from 6% to over 50%. Since the total number of end users in the topology is constant, as the number of attackers increases, the number of legitimate users decreases. All other parameters and experimental setup are consistent with the previous experiment. As before, for each mitigation algorithm, we perform 10 independent simulation runs.

In Fig. 7 we present the Interest satisfaction ratio for legitimate users aggregated over the 10 simulation runs for the first minute of the attack. The results are as expected—for all three mitigation algorithms, as the percentage of attackers in the network increases, the lower is the Interest satisfaction ratio for legitimate users. In the case of token bucket with per-interface fairness algorithm, just 3 attackers can halve the quality of service for the remaining 13 legitimate users. While the two intelligent attack mitigation algorithms also show a decline in service quality as the percentage of attackers increases, this decline is much more gradual and marginal. In the case of satisfaction-based pushback algorithm, during the first minute of the attack over 90% of Interests from legitimate users are satisfied even if 50% of end nodes are malicious.

B. Large scale simulations

In this section, we investigate the behavior of our mitigation strategies under a more realistic, large-scale network topology. The ISP-like topology we used is based on a modified version of Rocketfuel's AT&T topology [11]. We extracted the largest connected component comprising of 562 nodes from this original topology and separated the nodes into three categories: clients, gateways, and backbones. Nodes having degree less than four were classified as clients (344 red nodes as shown in Fig. 9), nodes directly connected to clients were classified as gateways (109 green nodes), and the remaining nodes were classified as backbones (109 blue nodes). We assigned bandwidth and delay values to links based on their type—both values are random numbers within the respective ranges as shown in Table I. We experimented with placing the data producer at both a gateway node as well as backbone node, which we randomly picked for each simulation run. Similar to the binary tree topology experiments, we fixed the number of malicious nodes at approximately 40% (140 out of 344 client nodes in the topology) and randomly picked these nodes for each simulation run. We conducted 10 simulation runs for each mitigation algorithm, with the attack duration spanning a 5-minute interval.

In Fig. 9, we summarize our results aggregated over all

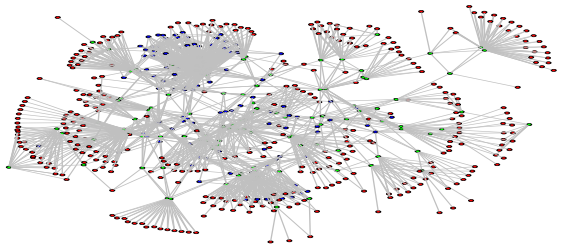

Fig. 8. Internet-like topology: 344 client routers (red), 109 gateway routers (green), 109 backbone routers (blue)

TABLE I
LARGE-SCALE TOPOLOGY LINK BANDWIDTH AND DELAY RANGES

Link type	**Delay**		**Bandwidth**	
	Min	Max	Min	Max
Backbone–Backbone	5 ms	10 ms	40 Mbps	100 Mbps
Gateway–Backbone, Gateway–Gateway	5 ms	10 ms	10 Mbps	20 Mbps
Client–Gateway	10 ms	70 ms	1 Mbps	3 Mbps

simulations runs for each mitigation algorithm for the scenario where the data producer is placed at a gateway node. We observe similar results for the data producer placed at a backbone node as well. Unlike the binary-tree topology experiments, we observe that both the token bucket with per interface fairness and satisfaction-based Interest acceptance have poor performance, while the satisfaction-based pushback is still the most effective algorithm. Interest satisfaction percentage for legitimate users are close to 30%, 25%, and 100% respectively for these mitigation methodologies.

Satisfaction-based Interest acceptance algorithm, which was very effective for binary-tree topology, is completely ineffective when deployed in a larger and more realistic topology. For the duration of the attack, legitimate users experience poor quality of service with only 25% of their Interests being satisfied and continue to experience degraded service long after the attack has stopped. This poor performance, as detailed in Section IV-A1, is due to the fact that each router on the path makes an independent, uncoordinated decision on whether to forward or drop an Interest. In the case of a large topology, with much higher average hop count, Interest packets from legitimate users have a very low probability of reaching the data producer, resulting in poor Interest satisfaction statistics and further penalization of new Interests from them.

All our evaluations leave us to conclude that among the three techniques we tested under various topology and attacker concentrations, the satisfaction-based pushback is the most promising one in mitigating Interest flooding attacks.

C. Limitations

This paper is a first step in understanding the impact of Interest flooding attacks in NDN and exploring the solution space. To design our mitigation algorithms we exploit two key features of NDN architecture, namely routers maintaining state about the Interests they have forwarded and Data traffic taking

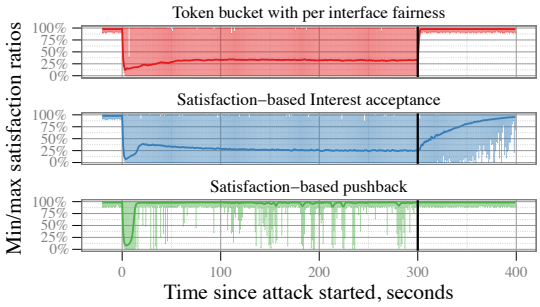

Fig. 9. Satisfaction ratio dynamics during the attack for large-scale topology with 40% attackers)

the reverse path of the Interest traffic. We test the efficacy of our algorithms by simulating them on both a simple binary-tree topology, as well as a more realistic ISP-based topology designed to provide insights of our algorithms' behavior when deployed on a real network. While our results are promising and show a great potential in mitigating Interest flooding DDoS attacks, there are certain limitations which should be addressed in future research.

First, in our evaluations we used a simple and static attacker model—attackers send junk Interests as fast as possible. In future work, we plan to explore the impact of models where the attackers are more sophisticated and dynamically adapt their behavior and Interest sending patterns based on the network reaction. Second, we assumed that Interests are not satisfied by an intermediate router's cache and always forwarded all the way to the producer. In future, we plan to study the impact of Interest flooding attacks in more realistic scenarios with multi-path routing enabled, more realistic traffic patterns, and the presence of in-network caches.

VI. RELATED WORK

As a new Internet architecture proposal, Named Data Networking has attracted a limited attention from the security community. Lauinger [12] showed several possible attacks against some specific design choices implemented in CCNx software (http://www.ccnx.org/); in particular [12] explored the issue of user privacy, assuming users are concentrated on the same edge router only and one can obtain complete knowledge of cached content. Wählisch et al. [13] explore security and stability threats in the general area of information-centric networks (ICN), using CCNx software as an example. While this work aims to generalize findings for all ICNs, it largely does not go beyond the current design choices of CCNx. In this paper we follow the suggestions by Gasti et al. [14] and utilize the properties provided by the NDN architecture to mitigate DDoS attacks, and tackle the Interest flooding problem as a first step in this direction. More specifically our attack mitigation design relies on NDN's stateful forwarding plane that allows us to maintain statistics on unsatisfied Interests. Similar approaches are also explored by Yi et al. [5], [6], [15] and Rozhnova et al. [16] to facilitate NDN network performance.

The general area of mitigating denial of service attacks is amongst the hottest topics in recent years. The most relevant works to this paper are the solutions to brute force IP packet flooding, such as DDoS detection techniques [17], [18], methods of pushing back malicious traffic to edges using various filtering techniques [9], [19], [20], [21], [22], [23], [24], overlay-based filtering [25], [26], [27], various ticketing systems with conditional admission of traffic in to the network [28], [29], [30], [31], [32], [33], and systems that attempt to use approximate IP traffic symmetry to estimate and filter out malicious traffic [34], [35], [36]. Identifying malicious traffic requires establishing certain state at routers, and the above mentioned solutions differ in the specifics on what state to use and how to establish it. By design NDN's forwarding plane keeps per packet state at every router, the finest granularity state to support any and all mitigation solutions—we simply take full advantage of this state to develop desired solutions.

VII. Conclusion

NDN being a newly proposed future Internet architecture, it is important to address its resilience in face of DDoS attacks. As an initial step in understanding the DDoS threats in NDN, 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 then leveraged the key features of the NDN architecture to design, develop, and evaluate three mitigation strategies. We performed detailed simulations on a range of topologies to quantify the effectiveness of our algorithms. The most effective algorithm—satisfaction-based pushback—was successful in almost completely shutting down the attackers, so that they cause little or no service impact to legitimate users.

NDN's stateful forwarding plane enables a number of desired functions, such as loop-free, multipath data delivery, built-in multicast, scalable content delivery, effective flow balance (i.e., congestion avoidance), and robust recovery from network failures, that people have attempted to install in IP networks [5], [6]. Although this useful per-packet state can be abused to launch attacks, the demonstrated success of satisfaction-based pushback algorithm serves as evidence that one can indeed utilize the per packet state built into each NDN router to enable effective DDoS mitigation as well.

References

[1] V. Jacobson et al., "Networking named content," in *Proc. of CoNEXT*, 2009.

[2] L. Zhang et al., "Named data networking (NDN) project," NDN Project, Tech. Rep. NDN-0001, October 2010.

[3] Arbor networks, "Worldwide infrastructure security report," http://www.arbornetworks.com/research/infrastructure-security-report, Volume VII, 2011.

[4] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *CCR*, vol. 34, no. 2, pp. 39–53, 2004.

[5] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *CCR*, vol. 42, no. 3, pp. 62–67, July 2012.

[6] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications: Information-Centric Networking Special Issue*, 2013.

[7] 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.

[8] 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.

[9] J. Ioannidis and S. M. Bellovin, "Implementing pushback: Router-based defense against DDoS attacks," in *Proc. of NDSS Symposium*, 2002.

[10] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," NDN Project, Tech. Rep. NDN-0005, 2012.

[11] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," *CCR*, vol. 32, no. 4, pp. 133–145, 2002.

[12] T. Lauinger, "Security & scalability of content-centric networking," Master Thesis, TU Darmstadt, 2010.

[13] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp, "Backscatter from the data plane—threats to stability and security in information-centric networking," *arXiv preprint arXiv:1205.4778*, 2012.

[14] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS & DDoS in Named-Data Networking," *arXiv preprint arXiv:1208.0952*, 2012.

[15] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," NDN, Tech. Rep. NDN-0002, 2012.

[16] N. Rozhnova and S. Fdida, "An effective hop-by-hop interest shaping mechanism for CCN communications," in *Proc. of INFOCOM WKSHPS*, 2012, pp. 322–327.

[17] Y. Chen, K. Hwang, and W. Ku, "Collaborative detection of DDoS attacks over multiple network domains," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1649–1662, 2007.

[18] J. Jun, H. Oh, and S. Kim, "DDoS flooding attack detection through a step-by-step investigation," in *Proc. of NESEA*, 2011.

[19] U. Tupakula and V. Varadharajan, "A practical method to counteract denial of service attacks," in *Proc. of Australasian computer science conference*, 2003, pp. 275–284.

[20] K. Argyraki and D. Cheriton, "Active internet traffic filtering: Real-time response to denial-of-service attacks," *USENIX*, 2005.

[21] G. Oikonomou, J. Mirkovic, P. Reiher, and M. Robinson, "A framework for a collaborative DDoS defense," in *Proc. of ACSAC*, 2006.

[22] X. Liu, X. Yang, and Y. Lu, "To filter or to authorize: Network-layer DoS defense against multimillion-node botnets," in *CCR*, vol. 38, no. 4, 2008, pp. 195–206.

[23] J. Chou, B. Lin, S. Sen, and O. Spatscheck, "Proactive surge protection: a defense mechanism for bandwidth-based attacks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1711–1723, 2009.

[24] X. Liu, X. Yang, and Y. Xia, "Netfence: preventing internet denial of service from inside out," in *Proc. of SIGCOMM*, 2010.

[25] R. Stone, "CenterTrack: An IP overlay network for tracking DoS floods," in *Proc. of USENIX Security Symposium*, vol. 9, 2000, pp. 199–212.

[26] A. Keromytis, V. Misra, and D. Rubenstein, "SOS: An architecture for mitigating DDoS attacks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 176–188, 2004.

[27] E. Kline, A. Afanasyev, and P. Reiher, "Shield: DoS filtering using traffic deflecting," in *Proc. of ICNP*, 2011.

[28] A. Yaar, A. Perrig, and D. Song, "SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks," in *Proc. of Symposium on Security and Privacy*, 2004.

[29] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," *CCR*, vol. 35, no. 4, pp. 241–252, 2005.

[30] D. Wendlandt, D. Andersen, and A. Perrig, "Fastpass: Providing first-packet delivery," Tech. Rep., 2006.

[31] M. Natu and J. Mirkovic, "Fine-grained capabilities for flooding DDoS defense using client reputations," in *Proce. of LSAD*, 2007.

[32] B. Parno, A. Perrig, D. Wendlandt, B. Maggs, E. Shi, and Y. chun Hu, "Portcullis: Protecting connection setup from denial-of-capability attacks," in *Proc. of the SIGCOMM*, 2007.

[33] T. Anderson, "Preventing internet denial-of-service with capabilities," *CCR*, vol. 34, no. 1, p. 2004, 2003.

[34] H. Wang, D. Zhang, and K. Shin, "Detecting SYN flooding attacks," in *Proc. of INFOCOM*, 2002.

[35] C. Kreibich, A. Warfield, J. Crowcroft, S. Hand, and I. Pratt, "Using packet symmetry to curtail malicious traffic," in *Proc. of HotNets*, 2005.

[36] A. Mahimkar, J. Dange, V. Shmatikov, H. Vin, and Y. Zhang, "dFence: Transparent network-based denial of service mitigation," in *NSDI*, 2007.