On Implicit Denial of Service Attack in NDN and Potential Mitigations

Gaurav Panwar, Reza Tourani, Travis Mick, Satyajayant Misra, Abderrahmen Mtibaa *
Department of Computer Science
New Mexico State University
{gpanwar, rtourani, tmick, misra, amtibaa}@cs.nmsu.edu

Abstract—One major benefit of named-data networking (NDN) is its potential to control network load by leveraging in-network caching and request aggregation. Both the network operator and consumers benefit from these features, as operating costs are reduced and quality-of-experience is increased.

However, request aggregation, combined with NDN’s loop prevention mechanisms, can create denial-of-service (DoS) against client interests (intentionally and unintentionally) by clients employing multicast forwarding. In this paper, we discuss this problem and propose three increasingly efficient solutions to address the problem; our arguments are backed by simulation and numerical analyses.

Index Terms—Information-centric networks; forwarding strategy; DoS attack; request aggregation.

1. INTRODUCTION

One major benefit of named-data networking (NDN) for the future Internet is its potential to help control network load in the face of increasing demand for high-bandwidth content. By leveraging in-network content caching and request aggregation, NDN helps reduce the number of content packets which transit the network without negatively affecting the consumers. Both the network operator and content consumer benefit from these features, as operating costs are reduced and quality-of-experience (QoE) is increased. The efficacy of both caching and aggregation in reducing network load can be attributed to the fact that most Internet content follows a Zipf popularity distribution [1], [2], and thus a small number of content objects represent the majority of consumers’ requests. There have been several efforts, studying the benefits of in-network caching and various approaches to optimizing its performance [3], however request aggregation has not received as much attention from the community.

To review, the main benefit of request aggregation is its reduction of redundant content forwarding through the network core. Request aggregation happens when a router receives an interest (request) for a content object for which an interest is already pending; instead of redundantly forwarding the interest (and thereafter receiving redundant data), the router aggregates the subsequent interests and satisfies them with the content returned for the original interest. Thus, a significant amount of bandwidth can be saved, particularly in live video streaming applications, where many consumers request the same content simultaneously.

Request aggregation has been identified to come at some cost, namely in information loss to content providers [4]. When consumers’ requests are satisfied from caches or by aggregation, the content provider gains no information about the consumer’s preference. However, in what follows we investigate a harmful side-effect of request aggregation that has been introduced in [5]: a denial-of-service (DoS) vulnerability. Other DoS attacks against NDN have been discussed in the literature [6], however, the authors in [5] have introduced a unique exploit against NDN’s request aggregation and loop prevention features, which could be triggered either unintentionally or maliciously by a client employing multicast forwarding. In addition, we propose and evaluate three increasingly-efficient solutions to the problem.

The contributions of this paper are: (i) A description of this unseen challenge inherent to request aggregation in NDN, which undermines successful content dissemination. (ii) A discussion of potential solutions and a discussion of their efficiency and cost. (iii) An implementation of the most viable solution and an analysis of its effectiveness through extensive simulation.

In Section 2, we describe the denial-of-service scenario we have identified and discuss situations in which it arises. Then, in Section 3, we propose and evaluate a naive approach to mitigating this attack. We refine our mitigation and present additional analysis in Section 4, and explore additional directions for improvement in Section 5. Finally, we offer concluding remarks in Section 6.

2. PROBLEM DESCRIPTION

We have identified a denial-of-service vector enabled by request aggregation, which is triggered by consumers multicasting interests. This behavior occurs due to NDN’s use of nonces for loop detection. Hereafter, we will refer to this unique NDN denial-of-service vector as iDoS.

Whenever an NDN application generates an interest, it assigns a random 32-bit integer, called the nonce. This nonce is used to determine the uniqueness of an interest, and as mentioned, mitigate forwarding loops in the network. When an NDN router receives a duplicate interest (same interest name and same nonce), it infers that there is a loop, drops the interest, and returns a negative acknowledgment (NACK). The NACK
Routers’ PITs in time

A

Provider

Router 1

Router 2

10 ms

15 ms

20 ms

Router 1

Router 2

(a) Network topology  (b) Routers’ PITs in time

Fig. 1: The case study for illustrating the request aggregation process under request broadcasting.

contains a flag indicating the reason for the interest’s rejection, which in this case is the duplicate nonce.

To illustrate iDoS, we will refer to the toy network depicted in Fig. 1(a), composed of a content provider, two routers (R1 and R2), and two consumers (A and B). Suppose both A and B request the same content object (d) simultaneously. For clarity, we use the client names A and B to represent the nonces for their respective interests. Consumer A, having two upstream interfaces, utilizes both by multicasting its request.

As shown in Fig. 1(b), at time step 10 ms, R1 receives a request for d from its face1 and R2 receives the same request over its face2. At time step 15 ms, R2 receives a new request for d over face1, and aggregates it with the entry it has in its Pending Interest Table (PIT). At the same time, R1 receives a redundant copy of the same request it received at time 10 ms, this time via face2. Since the request has the same content name and nonce, R1 detects it as a duplicate interest (depicted by the red shaded box in Fig. 1(b)). Due to its detection as a duplicate interest, R1 drops the interest from face2 instead of aggregating it with the existing entry, and sends a negative acknowledgment (NACK) to the downstream router via face2, indicating the receipt of a duplicate nonce. At 20 ms, R2 receives the duplicate-nonce NACK and removes the PIT entry for the request that it forwarded at time step 10 ms. In this scenario, consumer A receives content d at time 50 ms, while consumer B’s request times out despite the availability of d. This timeout occurs because no interest for d was successfully forwarded from R2 to R1. The first interest forwarded returned a NACK, and the second was aggregated prior to the receipt of that NACK. Therefore, R1 will not have forwarded d to R2.

To generalize this behavior, we define a multicast consumer as any consumer that utilizes more than one upstream interface when sending an interest. One can observe the effect of iDoS when (i) there is at least one multicast consumer co-located with other consumers, (ii) more than one consumer requests the same content object, (iii) the additional consumers’ requests are aggregated with a PIT entry that has been generated by a multicast consumer, and (iv) the multicast consumer’s interest is NACKed after the additional interests have been aggregated. It is also necessary that the multicast consumers’ request paths converge at some point; this provision will result in identical interests (with the same nonce) arriving at the point of convergence, and thus the generation of a duplicate-nonce NACK.

The iDoS attack can be triggered either unintentionally, by an honest multicasting consumer which happens to request the same content as the affected consumer, or intentionally by a malicious user who is capable of guessing which requests a target client will make. The unintentional attack is most likely to arise among consumers of live-streaming video traffic, as in these applications, many consumers concurrently request a content just prior to its generation by the publisher and thus request aggregation is more likely to occur. The real-time nature of live streaming media also makes it more likely for clients to employ multicasting forwarding strategies in order to meet QoE requirements.

Several strategies employing multicasting have been proposed in the literature already [7], [8], [9], [10], [11], however their effects when combined with interest aggregation have not yet been studied. With the exception of [11], which utilizes disjoint paths, all of the above cited forwarding strategies could potentially trigger the outlined DoS scenario. We argue that multicasting strategies, which may be installed on consumer nodes, cannot be safely deployed without augmentation by a multicast-aware forwarding strategy in the network core. Furthermore, even without the use of these strategies, it is possible for a malicious client to multicast interests and cause denial of service regardless; thus either interest aggregation must be disabled, or a mitigation must be employed.

3. Client-Driven Approach

In this section, we discuss a naive approach to mitigating iDoS, wherein consumer nodes utilize multiple nonces in order to prevent duplicate-nonce NACKs from being generated.

A. Description of Approach

In the multicast strategy currently available in the reference implementation of NDN, the node sends identical interests with the same nonce on all available faces. We propose a client-driven solution to prevent accidental iDoS of other clients in the network, wherein unique nonces are used on all outgoing faces. This strategy is henceforth referred to as Unique Nonce Multicast (UNM).

In UNM, the client node creates as many nonces as there are available faces, and then forwards the interest with a different nonce on each face. Though the consumer application is typically responsible for nonce generation in standard NDN, we suggest that the generation of additional nonces for UNM be handled by the strategy layer. This increases the implementation’s flexibility and facilitates compatibility with existing applications, as the application need not be aware of how many faces are available for forwarding. Because unique nonces are used, all of the multicast interests appear unique to the network and hence no duplicate-nonce NACK will be generated. Because there is no chance of an aggregating interest being NACKed, the iDoS scenario is avoided entirely.
the ideal case, the total number of deliveries will always be a non-empty subset of clients employing multicast forwarding. In scenario in terms of redundant data delivery in a situation where redundancy is chosen to analyze the case of optimum value of clients.\(\alpha\) have three interfaces.\(\alpha = 300\) T seconds of activity, where all multicasting clients \(\alpha \in \{40, 80, 120\}\) clients and with \(\alpha \in \{1, 0.75, 0.5, 0.25\}\). For purpose of visualization, we have chosen to analyze the case of \(r = 10\) interests per second with \(T = 300\) seconds of activity, where all multicasting clients have three interfaces.\(\alpha = X_3\). Note that the number of data packets delivered increases as the fraction of multicasting clients \(\alpha\) increases; when \(\alpha = 1\), the result matches the optimum value of \(rT|\mathcal{X}|\).

This behavior of \(UNM\) actually represents a worst-case scenario in terms of redundant data delivery in a situation where a non-empty subset of clients employ multicast forwarding. In the ideal case, the total number of deliveries will always be

\[
 rT \left( \sum_{i \geq 1} i |X_i| \right).
\]

(3.1)

In Fig. 2, we show the result of our numerical calculations regarding the number of content objects delivered to clients in networks containing \(X \in \{40, 80, 120\}\) clients and with \(\alpha \in \{1, 0.75, 0.5, 0.25\}\). For purpose of visualization, we have chosen to analyze the case of \(r = 10\) interests per second with \(T = 300\) seconds of activity, where all multicasting clients have three interfaces.\(\alpha = X_3\). Note that the number of data packets delivered increases as the fraction of multicasting clients \(\alpha\) increases; when \(\alpha = 1\), the result matches the optimum value of \(rT|\mathcal{X}|\).

This behavior of \(UNM\) actually represents a worst-case scenario in terms of redundant data delivery in a situation where a non-empty subset of clients employ multicast forwarding. In the ideal case, the total number of deliveries will always be

\[
 rT|\mathcal{X}|,
\]

as each request results in exactly one content delivery. When employing standard multicast strategies such as those mentioned in Section 1, the number of content packets delivered will be bounded below by \(rT|\mathcal{X}|\) and above by Equation 3.1.

Though redundant content delivery indeed burdens the network, \(UNM\) also has a much more serious problem: it offloads the responsibility of mitigating \(iDoS\) onto clients. In addition to the impossibility of ensuring that all legitimate multicasting clients use \(UNM\), there is no way to prevent a malicious client from choosing not to install it. Therefore, in the next section we propose an alternate mitigation to \(iDoS\) which does not depend on the behavior of clients, and instead employs an in-network reactive approach.

### 4. In-Network Approach

In this section, we discuss a mitigation to \(iDoS\) which employs in-network retransmissions of NACKed interests to ensure content delivery. Different from [12] where in-network retransmissions are utilized to avoid loop-detection NACKs to allow face probing, we use in-network retransmission to ensure delivery to all clients in the network.

#### A. Solution Description

To mitigate \(iDoS\) without relying on the cooperation of all clients, we propose a modified forwarding strategy meant to be deployed in the core of the network. This strategy is henceforth referred to as Request Re-Transmit (\(RRT\)). Under \(RRT\), routers may retransmit interests which are NACKed due to duplicate nonces, and thereby ensure that all consumers receive the content objects they have requested.

Algorithm 1 details the procedure followed by an in-network router implementing \(RRT\) upon the receipt of a duplicate-nonce NACK for interest \(k\). On receiving the NACK, the router checks the PIT entry associated with the NACKed interest to determine whether any additional interests \(k' \in K\) were aggregated with it (Line 1). If such an interest exists, it will have a different nonce associated with it. Therefore, the router can retransmit that aggregated interest back on the face on which it received the NACK (Lines 2-8). Finally, the router forwards the NACK
downstream on the face from which it originally received the NACKed interest (Lines 10-11).

Fig. 3 illustrates a working example of this technique. As in the earlier example, both clients A and B request d simultaneously, and A multicasts its interests. In this example, both Router 1 (R1) and Router 2 (R2) are using RRT. For clarity, we again use the client names (A and B) to represent the nonces for their interests. As shown in Fig. 3(b), at time step 10 ms, R1 receives a request for d from its face1 and R2 receives the same request over its face2. At time step 15 ms, R2 receives a new request for d over face1, and hence aggregates it with the entry it has in its PIT. At the same time, R1 receives the same request (identical nonce) it received at time 10 ms from its face2. Since the request has the same content name and nonce value, R1 detects it as a duplicate request (red shaded entry in Fig. 3(b)). Due to the request being a duplicate, R1 did not add face2 to its existing PIT entry for the request for content d, but instead sent a NACK down on face2 towards R2.

At 20 ms, R2 receives the duplicate NACK and detects that it has an aggregated interest for face1. R2 retransmits the interest with the nonce associated with face1 up to R1. It also removes face2 from its PIT and forwards the NACK down on face2 (orange shaded entry). When R1 receives data at time step 40 ms, it would forward it on both face1 and face2. R2 forwards data only on face1. In this scenario, consumer A receives content d at time 50 ms via face1, and consumer B receives data at time 60 ms via face1. As illustrated in this example, RRT ensures that no client suffers packet loss due to iDoS.

### B. Simulation Results and Analysis

Now, we present the performance analysis of RRT based on ndnSIM [13] (a module of ns-3).

We evaluated RRT on scale free networks generated according to the parameters in Table I. A given number of core routers were created, and some subset of them were then reassigned to be edge routers. A content provider was then attached to one of the edge routers; then, clients were attached to the edge routers deemed distant enough from the producer. In the 200- and 600-node network, the distance threshold was four hops; in the 400- and 600-node networks, the threshold was five hops. Unicast clients were each randomly given a single link to an edge router, and multicast clients were each given three random links to edge routers (while ensuring that no client is connected to the same edge router more than once). All client-to-edge-router links had 1 Mbps bandwidth with 10 ms propagation delay, and the core of the network had links with 10 Mbps bandwidth with 1 ms delay. We implemented RRT by modifying the BestRoute2 forwarding strategy available in ndnSIM according to Algorithm 1. We compared the performance of BestRoute2 (BEST) with that of RRT when deployed in the network core. All clients requested the same sequence of content objects from the producer at a rate of ten interests per second. The interest lifetime for the clients was set at 100 ms. We ran our simulations for 300 seconds and averaged the results for each topology over ten runs with distinct seeds.

We will evaluate the performance of RRT compared to BEST considering the following criteria: (i) Success rate (the ratio between the number of data packets received by the client and the number of interests), and (ii) Overhead (total number of excess interest and NACK packets seen in the network).

Fig. 4(a) shows the success rate of RRT compared to BEST. The wide variation in the success rate for BEST, reflected by its error bars, shows the non-deterministic nature of iDoS and its dependence on the network topology. However, RRT ensures that no requests are lost due to duplicate nonces; thus, 100% of requests are satisfied.

Fig. 4(b) shows the number of interests observed in the network. In each of the topologies, RRT shows higher interest overhead when compared to BEST, due to retransmissions.

Fig. 4(c) shows the number of NACKs observed in the network. This figure shows that RRT results in fewer NACKs being propagated within the networks. In case of BEST, the clients which time out due to NACK retry the interests and compete for same data again which might cause more iDoS scenarios in the network resulting in more NACKs seen in the network.

Fig. 4(d) shows the total number of content objects delivered to clients, including redundant deliveries. As discussed in Sec. 3.2, redundant delivery is nearly inevitable when clients employ a multicasting strategy. Under BEST, a NACK results in an aggregated interest’s entire delivery tree being pruned; however, RRT intentionally avoids this pruning step in order to increase content delivery success. Thus, in addition to increased satisfaction of interests which would have been lost under BEST, we also observe an increased rate of redundant delivery.

In addition to these analyses, we also measured content...
delivery latency in BEST and RRT. However, we observed that delivery latency under RRT follows the same distribution as delivery latency under BEST. This result is attributable to the fact that in the topologies we studied, the total latency of the NACK generation and interest retransmission process is less than the content retrieval latency from the NACKing node to the content provider. Therefore, the PIT state in the upstream router will be corrected prior to it receiving the content object, and no additional delay is induced by the retransmission process. We also measured the number of retries required in the cases where RRT received a NACK, and determined that 96% to 98% of retransmitted interests are only retransmitted once. The remaining 2%-4% of cases where more than one retry occurs arise in cases where more than two requests are aggregated by the same router and more than one of the corresponding nonces is NACKed.

The results discussed above show that RRT is able to successfully and transparently mitigate iDoS, as it has satisfied 100% of requests in the evaluated scenarios without impacting client-observed latency. Though RRT reduces NACK overhead in the network, it does induce additional interest transmissions compared to BEST, and more importantly, exhibits a higher rate of redundant content delivery. Our goal with RRT was not to optimize for network overhead, but rather to ensure that user QoE is not impacted by iDoS. However, in the following section we discuss potential directions for improvements to RRT which would reduce this overhead.

5. Discussion

The proposed RRT forwarding strategy leaves room for improvement in terms of interest and redundant content delivery overheads. As mentioned, a small portion of requests must be retried multiple times, due to the existence of multiple levels of aggregation in the network and multiple multicasting clients. If these additional retries are prevented, some network load can be reduced, and potential boundary cases where delivery latency is increased can be eliminated. Also, some amounts of redundant content delivery occurs naturally when clients multicast their requests. This overhead has a larger impact on the network. In this section, we provide a brief discussion of improvements to RRT that would mitigate both of these sources of overhead.

Fig. 5 illustrates an example of multiple retries and content deliveries. Clients A, B, C, and D request the same content d at the same time, with A, C, and D multicasting their interests. Both routers Router 1 (R1) and Router 2 (R2) are using RRT. We illustrate the PIT of each router in Fig. 5(b) with the face it receives the interest message on and the nonce of the interest generated by the client. For clarity, we use the client names, A, B, C, and D to represent the respective nonces in their interests. In Fig. 5(b), at time step 10 ms, R1 receives a new
request for \(d\) over its \(face1\) from \(C\) and \(R_2\) receives the new request over its \(face3\) from \(A\). Both \(R_1\) and \(R_2\) create PIT entries and forward the data towards the producer (\(face4\) for \(R_1\) and \(face4\) for \(R_2\)).

At time step 15 ms, \(R_1\) receives a new request over \(face3\) from \(A\) and hence aggregates it with the entry it has in its PIT. At the same time, \(R_2\) receives a new request over \(face1\) from \(C\) and aggregates it into its PIT. At time step 20 ms, \(R_1\) receives a new request over \(face5\) from \(D\) and aggregates it into its PIT. \(R_1\) also receives a duplicate request (duplicate nonce) on \(face2\) for a content which matches the nonce \(A\) received on \(face3\). \(R_1\) does not aggregate this interest and sends a duplicate NACK down on \(face2\) (shown in red shaded entry). At the same time step, \(R_2\) receives a new request over \(face2\) from \(B\) and aggregates it into its PIT.

At time step 30 ms, \(R_2\) receives the NACK on \(face4\) matching nonce \(A\) for its entry for \(face1\). \(R_2\) removes the PIT entry for \(face3\) and sends NACK down on \(face3\) (shown in orange shaded region). \(R_2\) also retransmits interest with nonce \(B\) up \(face4\) as part of the NACK handling operation of \(RRT\). \(R_2\) also receives a new request over \(face5\) from \(D\) and aggregates it into its PIT. At time step 40 ms, \(R_1\) receives a duplicate request on \(face2\) for content which matches nonce \(C\) \((face1)\). \(R_1\) does not aggregate this interest and sends a duplicate NACK down on \(face2\) (shown in red shaded entry).

At time step 50 ms, \(R_2\) receives the NACK on \(face4\) matching nonce \(C\) for its entry for \(face1\). \(R_2\) removes the PIT entry for \(face1\) and sends NACK down on \(face1\) (shown in orange shaded region). \(R_2\) also retransmits interest with nonce \(B\) up \(face4\) as part of the NACK handling operation of \(RRT\). At time step 60 ms, \(R_1\) receives a new request over \(face2\) with nonce \(B\) and hence aggregates it with the entry it has in its PIT. At time step 70 ms, \(R_1\) would receive data from \(face4\) and forward it down on \(face1\), \(face2\), and \(face3\). At time step 80 ms, \(R_2\) would receive data from \(face4\) and forward it down on \(face2\). In the given example all clients received data (Client \(A\) at 85 ms, Client \(B\) at 100 ms, Client \(C\) at 80 ms, and Client \(D\) at 90 ms and 110 ms), and \(R_2\) had to retransmit interests two times on \(face4\) (at time step 30 ms and 50 ms).

This example shows that in a large network these kind of retries and redundant content deliveries (e.g., to \(D\)) could end up being significant. To reduce these redundancies, there needs to be a modification at the router \((R_i)\) which initiates the NACK due to receipt of duplicate interests. For example, a potential modification is for \(R_1\) to include all the nonces in the aggregated PIT entry in the NACK for a duplicate interest. This would give more information to downstream routers (closer to clients) when they receive the NACK. For example, at time 20 ms, \(R_1\) could add its set of nonces \(\{A, C, D\}\), with the NACK. That would prevent \(R_2\) from sending the interest for \(C\) again at time 25 ms and retry with interests for \(B\). In turn, \(R_2\) can NACK back to \(A, C,\) and \(D\). This prevents redundant data delivery to \(D\).

If the downstream routers have interests aggregated with nonces different from the list of nonces included in the NACK, they can choose to retransmit the interest which will get aggregated at \(R\) without multiple retries. If the nonces of aggregated interests from downstream routers are included in the NACK, then downstream routers will not redundantly retransmit interests up towards the producer. This solution will decrease the number of in-network interest retransmissions and redundant data deliveries resulting in reduced network overhead.

This solution requires modification to the structure of NACKs as well as the forwarding behavior of nodes. Due to lack of time, we have not implemented this solution in ndnSIM; the modifications require changes in many facets of the NFD.

### 6. Conclusions

In this paper, we demonstrated the scenario where multicasting clients in NDN networks can end up staging an implicit DoS attack on other clients. We discussed three solutions, \(UNM\), \(RRT\), and an improvement to \(RRT\), to mitigate the implicit DoS scenario. We implemented \(RRT\) and showed that it can successfully prevent the implicit DoS of clients. We did not implement the improved version of \(RRT\) discussed in Section 5 as it needs modification to the core ndnSIM structure and NACK packets, which we will explore in the near future.

### References


