IBPC: An Approach for Mitigation of Cache Pollution Attack in NDN using Interface-Based Popularity

Naveen Kumar (nk10121989@gmail.com)
Siksha O Anusandhan University https://orcid.org/0000-0003-0882-3119
Shashank Srivastava
Motilal Nehru National Institute of Technology

Research Article

Keywords: Cache Pollution Attack, NDN, Named Data Networking, NDN Security

Posted Date: July 12th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-682924/v1

Noname manuscript No.
(will be inserted by the editor)

IBPC: An Approach for Mitigation of Cache Pollution Attack in NDN using Interface-Based Popularity

Naveen Kumar · Shashank Srivastava

Received: date / Accepted: date

Abstract The performance of Named Data Networking (NDN) depends on the caching efficiency of routers. Cache Pollution Attack (CPA) refers to colonization of unpopular contents in the Content Store (CS) of an NDN router, which leads to declined Quality of Service (QoS) in NDN. CPA has very few solutions proposed for its mitigation. Most of these solutions are based on the statistics of the router itself. However, an attacker can influence these statistics by requesting unpopular contents repeatedly. This article proposes a new parameter for the detection of CPA, which is based on the number of distinct users requesting interest packets for a content over a period of time. The local popularity of the attackers' content does not affect the proposed approach. Results show that the proposed approach consumes less storage, reduces processing time, and more effectively mitigates the CPA, as compared to the other existing approaches.

Keywords Cache Pollution Attack; NDN; Named Data Networking; NDN Security

1 Introduction

TCP/IP was developed as a solution for communication among hosts. With the evolution of the Internet, the usage of TCP/IP shifted from communication to content sharing. Online gaming and services like Netflix, YouTube, Facebook have become popular now. The global IP traffic per month is expected to reach 194 Exabytes, and the IP video traffic is expected to be 82% of all traffic by the year 2020 [21]. TCP/IP model, which was initially designed to solve the

N. Kumar
Shiksha 'O' Anushandhan University, ITER
E-mail: naveenkumar@soa.ac.in

S. Srivastava
Motilal Nehru National Institute of Technology Allahabad

problem of communication among the hosts, is now being overwhelmingly used for content sharing. Many solutions, viz., Peer-to-Peer (P2P) network, Distributed Hash Table (DHT), and Content Delivery Network (CDN), have been developed to make TCP/IP capable for sharing contents. However, these solutions are not optimal as the transmission of packets is delayed due to the underlying network. For solving such problems, new types of networks have been introduced, which are content-centric rather than host-centric. These networks come under the umbrella of Information-Centric Networking (ICN). Few examples of ICNs are Data-Oriented Network Architecture (DONA) [14], COntent Mediator architecture for content-aware nETworks (COMET) [9], Content Centric Networking (CCN) [11], and NDN.

NDN is the most outstanding candidate for future network architecture among all the ICNs. NDN solves the problem of content sharing by giving unique names to each content and facilitates content fetching through these allocated names. Naming in NDN solves many issues such as mobility, DNS overhead, and user anonymity. NDN also facilitates content caching within the router. The caching ability of the router improves the performance of NDN by minimizing the average delay in content retrieval. However, it opens a door for an attacker to access the router's cache. An attacker can deliberately request unpopular contents that get cached in NDN routers. These contents are unpopular; therefore, they are rarely requested by other users. The increase in the number of unpopular contents in the CS leads to a decline in the hit ratio of the router's cache. This attack is known as cache pollution attack.

CPA is very difficult to detect, unlike other NDN attacks, like Interest Flooding Attack [1] and Cache Privacy Attack [16]. The attacker does not need to follow any specific pattern for performing CPA. Deng et al. [6] have given two variants of CPA– Locality Disruption Attack (LDA) and False Locality Attack (FLA). In LDA, the attacker requests a lot of new unpopular contents. These unpopular contents get cached in the gateway and its nearby routers. These cached contents are unpopular; therefore, they are rarely requested by a legitimate consumer. Hence, the hit ratio of the requests generated by legitimate consumers decreases. Most of the countermeasures [17, 22, 4, 13, 23, 12, 18, 25] employed for mitigating LDA are caching schemes based on the local popularity of contents. These schemes calculate the popularity of contents based on the statistics of the router.

In FLA, the attacker manipulates the popularity of already cached contents by requesting a set of unpopular contents repeatedly. FLA increases the overall hit ratio of the cache as the attacker requests the same set of contents repeatedly. However, the hit ratio of legitimate consumers decreases as some portion of the cache is occupied by the unpopular contents which are requested by the attacker repeatedly. Therefore, local popularity based approaches favor these unpopular contents as they have high popularity. Thus, the local popularity-based caching schemes are ineffective against FLA. A trivial solution for FLA is using global popularity as a metric for caching the content. However, calculating global popularity and maintaining the popularity state on each router incurs much overhead.

This article proposes an Interface-Based Popularity Caching (IBPC) approach, which uses a new parameter for the mitigation of LDA and FLA. The IBPC approach caches the content according to the number of interfaces that receive the content in a given period. The basic idea behind this approach is that the popularity of contents is proportional to the ratio of number of interfaces on which the content is received to the total number of interfaces. The rest of the article is organized as follows. Section 2 briefs the NDN architecture. Section 3 reviews the related work. Section 4 presents CPA and its execution in NDN. Section 5 discusses the proposed approach - IBPC along with the mathematical analysis for storage and computational overhead. Section 6 presents the experimental results and the comparative analysis of IBPC with the CPMH approach [12], and the approach discussed by Xie et al. [22]. The paper is concluded in Section 7. The significant contributions of this article are as follows:

1. The literature reveiw comparatively analyse the various metrics used for CPA in the existing approaches.
2. Execution of an efficient LDA and FLA in NDN, using some contents.
3. Proposal of a novel technique for mitigation of CPA along with the estimation of the storage and processing overhead.
4. Implementation of the IBPC approach along with existing approaches, which mitigates LDA and FLA.
5. Comparative analysis of the IBPC approach and the other existing approaches.

2 NDN Architecture

NDN employs a human-readable hierarchical name structure, with each name containing name components separated by a delimiter of "/". For example, */BBC/News/MorningNews.mp4*. NDN uses two types of packets for a communication, i.e., the data packet and interest packet. The data packet serves as a response to the interest packet, which is used to request information.

Pending Interest Table (PIT), Forwarding Information Base (FIB), and Content Store (CS) are the three types of data structures used by NDN router. PIT is used to store unsatisfied interest packet's metadata (i.e., name and incoming interface list) before the matching data packet is received or the timer for that entry expires. Replacement policies like Least Recently Used (LRU), Least Frequently Used (LFU), etc., are used by CS to cache data packets. The FIB is used to determine the interface using which the received interest packet should be forwarded. Figure 1 depicts the forwarding of the interest packet and the data packet.

When an NDN router receives an interest packet, it first searches the CS for the matching data packet. If the data packet is located in the CS, it is returned to the sender. Otherwise, the router performs a PIT lookup. The incoming interface is added to the interface-list of the matching PIT entry if a matching entry is identified. Otherwise, a lookup is performed using the

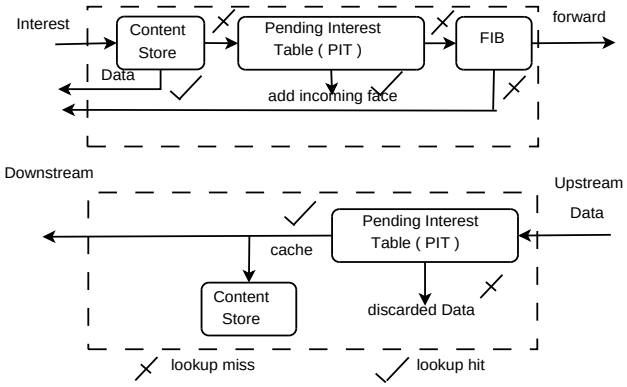


Fig. 1 Forwarding pipeline in NDN

interest packet's name in FIB based on the longest matching prefix. In the PIT, an entry is made; if there is a matching entry with the interest packet and the incoming interface. The interest packet is also forwarded from the interface specified by the FIB entry. Otherwise, depending on the router's forwarding policy, the interest packet is dropped or a negative acknowledgment is sent.

When an NDN router receives the data packet, then it looks for the matching PIT entry. If the entry is found, then the data packet is forwarded through all the interfaces present in the matching PIT entry; additionally, the data packet is cached in CS. Else, the data packet is dropped, or a negative acknowledgement is sent deeding upon the policy.

3 Related Works

Pollution attacks are not new as they have already been studied in the context of web caching on proxy servers in [7] [8]. This section mainly focuses on CPA in NDN.

Park et al. [17] proposed an approach for detecting LDA based on calculating the entropy of requests. This approach is based on mapping a content object to an entry in a binary matrix, where the indices of the matrix are obtained by hashing the content's name and taking modulus by using two different positive integers. If the object is present, then the corresponding entry is set to 1; else, it is set to 0. Gaussian elimination is applied for calculating the rank of the matrix. When this rank value goes beyond a predefined threshold, then the attack is detected. The authors have not applied a countermeasure after the detection. This approach uses a lot of CPU cycles per request for calculating the hash.

Xie et al. [22] proposed a caching approach for the mitigation of LDA, which is based on computing a shield function. The cache stores the placeholders for the contents which are received but not cached along with the actual contents. This placeholder contains the name of the content and its count. The count

represents the number of times the content is requested. When a router receives a content, it increments the count and computes the shield function.

$$\psi(t) = \frac{1}{1 + e^{(p-t)/q}}, t = 1, 2, ... \quad (1)$$

Here t is the t^{th} request for a given content object in CS, and p and q are the parameters of the function. The authors have used p, and q equals to 20 and 1, respectively. If the shield function's value goes beyond a predefined threshold, then the router decides to cache the content. This approach uses a larger space because the router has to store placeholders along with the contents. Also, it consumes critical CPU cycles of the router for calculating shield function whenever a content is received.

Conti et al. [4] proposed an approach for detecting LDA and LFA simultaneously. learning phase and the identification phase are the two stages of this strategy. The caching algorithm generates a set S during the learning process by randomly sampling IDs of contents. The router keeps track of S's contents in order to determine the *tau* threshold, which is used to detect the attack. During the detection process, the router calculates the measurement variability, or $delta_m$, which is dependent on the frequency of content that belongs to S, the measurement size, and the likelihood of content occurrence. The attack is detected if $delta_m$ is greater than *tau*. Since the router only needs to keep statistics for a limited subset of the contents cached in CS, this solution uses less memory and computation. After the detection, the authors have not given any mitigation approach.

Karmi et al. [13] proposed a cache replacement policy that is based on an Adaptive Neuro-Fuzzy Inference System (ANFIS). Six different input parameters were considered for training the network. The caching decision is taken based on the trained ANFIS-based neural network.

Xu et al. [23] proposed a countermeasure for LDA and FLA, which is based on the assumption that for a small group of unpopular namespaces, the attacker demands a large number of interest packets. It is divided into two phases: traffic monitoring and identification. In the traffic monitoring process, the Lightweight Flajolet Martin (LFM) sketch is used to track distinct interest packets with the same namespace. Monte Carlo hypothesis test [10] is used to calculate the attack detection threshold. The router tracks traffic on a regular basis during the detection process and calculates the interest-traffic monitoring result. If this value exceeds the empirical threshold, the router concludes that an attack is underway. However, an attacker can avoid detection by performing the attack using unpopular contents from different namespaces.

Kamimoto et al. [12] proposed an approach to mitigate CPA using the hierarchy of contents. There are three steps to this method: 1) determining the prefixes used by the attackers, 2) cache recovery, and 3) cache protection. The router calculates the Weighted Request Rate Variation per Prefix (WRVP) in the first step, which is used to create a blacklist. The router then removes cached content from the black-list in the second step. The router does not cache future unpopular contents in the third step. Since namespace statistics

Ref.	Attack Type	Popularity Matric	Memory Overhead	Computational Overhead
Park et al. [17]	LDA	Entropy-based Detection	Low	High
Xie et al. [22]	LDA	Shield Function	High	High
Conti et al. [4]	Both	VOM	Low	Low
Karami et al. [13]	Both	ANFIS-based Decision	Low	Low
Xu et al. [23]	Both	LFM Sketch	Low	Low
Kamimoto et al. [12]	FLA	WRVP	Low	Low
Salah et al. [18]	Both	RR	High	Low
Zhang et al. [25]	LDA	COV and PPP	Low	Low
Zen et al. [20]	FLA	Four parameters	High	Low
Singh and Ujjwal [24]	Both	Interest popularity	High	Low

Table 1 Comparative Analysis of the Various Metrics Used for CPA

are stored per namespace rather than per content, this approach uses less memory than the other approaches [17] [22]. This approach has the drawback that an attacker can use the unpopular contents' popular prefix to carry out the attack. Since these interest packets are associated with a popular prefix, they are not included in the blacklist.

Salah et al. [18] utilized the CoMon framework for the mitigation of CPA. The nodes are classified into three categories– ISP Controller (IC), Monitoring Nodes (MNs), and NDN Nodes (NNs). The MNs periodically send the IC a list of contents with the number of times they are requested. IC creates a white-list based on the Request Rate (RR) of each content. RR is the number of times a content is requested in a period. The white-list contains prefixes that are requested frequently. Contents in the white-list are distributed among MNs. The MN caches contents which are present in the white-list. The NNs do not cache the contents present in the white-list of MR to which the NNs are associated. This approach ensures that the popular data always present in the cache of the MNs, which reduces the number of costly Inter-ISP data requests. This approach uses a complex architecture over NDN, which incurs extra message overhead and consumes extra CPU cycles of the router.

Zhang et al. [25] proposed an approach for mitigating CPA that is dependent on the content's location. Two criteria are used to determine whether or not to cache: the coefficient of variation (COV) and the popularity per prefix (PPP). The router updates its data set when it receives an interest packet, which includes information such as the prefix, incoming interface, and frequency of the interest packet; this data set is used to calculate COV. The popularity of a prefix P is measured by the number of times interest packets with P as a prefix are requested. When the router receives a data packet, it decides whether or not to cache it based on the popularity and COV values.

Zen et al. [20] have proposed an approach for the mitigation of FLA using grey forecast. This approach predicts future popularity by using the request ratio, the variance of repeated interests, the standard deviation of request intervals, and Content Popularity. The actual popularity of content is computed and compared with the predicted popularity. If the deviation is large, FLA

is detected, and a filter is applied to show that malicious content will not be cached.

Singh and Ujjwal [24] have proposed an approach for the mitigation of cache pollution attack based on Gini impurity. The Gini impurity of the interest packet is computed using interest's probability. This newly computed Gini impurity value is compared with the Gini impurity of the previous time slice. Based on this difference, the attack is detected. The information of the malicious interest packet is propagated to downstream routers. These routers then do not cache the malicious content.

In the Table 1, various mitigation approaches are compared based on their effectiveness against LDA or FLA, popularity matric used for caching, memory overhead, and computational overhead. The popularity of matric(s) used by individual countermeasures is used to accept or reject the content. The memory and computational overhead is the relative memory space and CPU cycle needed for deploying the countermeasure. All the approaches are based on the popularity of the content measured using local information of the router except the Salah et al. approach [18]. The initial assumption of Xu et al. [23] that the attacker has to request a large number of interest packets for a small group of unpopular namespaces for performing CPA is not necessary. However, LDA or FLA can be performed using a few contents, as it has been illustrated in section 4 of this article. Karami et al. approach [13] is based on a trained Neuro-Fuzzy network, which depends on the type of attack, topology, traffic, etc. This approach entirely depends on the local popularity of the content, which can be influenced by the content requested by the attacker. Conti et al. approach depends on a small set of contents that are chosen initially. Therefore, the router can monitor only the contents which are present in the set. Also, the authors do not mention any specific way of choosing a set of contents from the CS, which can benefit detection. Thus, there is no effective countermeasure for the mitigation of LDA and FLA both. The IBPC approach does not get influenced by the false popularity created by the attacker in FLA or a large number of unpopular contents requested by the attacker in LDA.

4 CPA: An Insight

In CPA, the attacker intends to reduce the cache's performance by requesting unpopular contents, which results in the decrement of the hit ratio of the legitimate consumers. Types of CPA have already been discussed in Section 1. This section discusses the actual execution of both types of attacks in NDN.

4.1 Executing Efficient LDA

In LDA, the attacker intends to decrease the hit ratio of the consumers' request by requesting new unpopular contents. It may seem that LDA can be easily executed by requesting a large number of unpopular contents one by one.

However, the router can easily detect this attack by observing the number of times a content is requested. If any requester does not request a content for a threshold time, then it can be evicted. Thus this attack can be easily detected. To make the attack more difficult to be detected the attack should be performed using a set of a small number of contents. In the first view, it may seem that many unpopular contents are needed for executing LDA. However, it is not necessary as unpopular cached contents eventually get evicted due to the caching of new contents requested by other consumers. After the eviction of the contents cached by the attacker, it can be reused for the attack. Thus, if the attacker knows a rough estimate of the upper bound of the time (t_u) for which an unpopular content remains in the cache, then the attacker can reuse the previously requested contents for performing the attack.

Lauinger et al. [15] have coined the term - 'characteristic time' for the approximate upper bound of a content's residing time in the cache. The authors have also proposed an approach to calculate the characteristic time (t_u), which is shown in Figure 2. Initially, the attacker intentionally inserts n unpopular contents, i.e., O_1, O_2, ... , O_n in the cache at time t_0. Then, the attacker requests O_1 after $\frac{1}{f_p}$ time period, O_2 after $2 * \frac{1}{f_p}$ time period, and so on until a hit occur, here f_p is probing frequency. Suppose hit occurs at t_h when h^{th} content is requested then t_u is given by equation below

$$t_u = t_h - t_0 \tag{2}$$

Now, suppose the attacker wants to ensure that k contents belong to set S must remain in the cache, then the attacker has to request these contents during the t_u time. Thus the attack frequency will be k/t_u. However, if the attacker requests the content with a fixed frequency, then it can be easily detected by the router.

For making the attack more realistic and in-detectable, the attacker can mimic the consumer by requesting content after a t time where t is obtained using an exponential distribution. Thus, LDA can be performed using Algorithm 1.

Algorithm 1 Efficient LDA

INPUT: $numIteration$, k, t_u

```
for i = 0; i < numIteration; i++ do
    C = chooseRandom(S)
    request(C)
    t = Expo(k/t_u)
    sleep(t)
```

It can be seen that the content is chosen randomly from the set S. It makes the pattern of requesting the content random. LDA has a trait that the popularity of contents that the attacker requests is very low; therefore, LDA mitigation is easier.

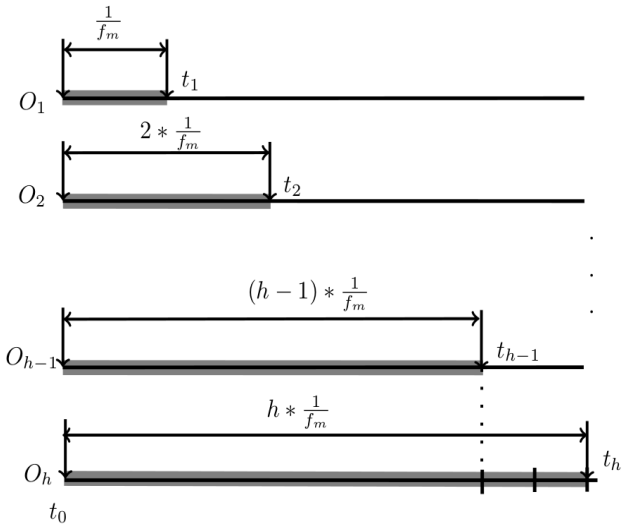


Fig. 2 Finding t_u

4.2 Executing Efficient FLA

In FLA, the attacker increases the popularity of a set of contents by repeatedly requesting them. Thus, local popularity-based approaches fail against FLA as the attacker makes the unpopular contents popular intentionally. Suppose the attacker knows the popularity threshold (l), i.e., the number of requests needed in time (t_p) to make an unpopular content popular. If the attacker wants to ensure that k contents must be cached, it must request each content l times within t_p time. Thus FLA can be performed using Algorithm 2.

Algorithm 2 Efficient FLA

INPUT: $numIteration$, l, k, t_p

```
1: for i = 0; i < numIteration; i++ do
2:     C = chooseRandom(S)
3:     request(C)
4:     t = Expo(t_p/(k × l))
5:     sleep(t)
```

5 Proposed Work: IBPC

In Section 1, it has been discussed that local popularity (popularity calculated using statistics of the local router, like the number of requests per content)

based caching is effective against LDA but is in-effective against FLA. Measuring global popularity is difficult due to message overhead in maintaining the popularity state. Also, ignoring the content's previously calculated popularity ratings may lead to wide fluctuations in hit ratios and may fail against sudden high-frequency attacks. The IBPC approach has been designed to overcome these limitations. It is based on the assumption that the number of attackers in a network is less than the number of consumers. This assumption is obvious in all the realistic attack scenarios. The number of distinct users requesting the content becomes a significant factor in deciding the content's popularity. Hence, the IBPC approach requires the routers to calculate the number of distinct users who request a given content in a period.

The IBPC approach utilizes the information related to number of content requesters for calculating the current popularity of the content. This newly calculated rating is passed through the Exponentially Weighted Moving Average (EWMA) [5], which is used to calculate each content's subsequent popularity. Let the total numbers of users (legitimate users and attackers together) in the network be n, the number of distinct users requesting the content C in the current period T_i be x_i, then the new rating R_i of content for this interval is calculated by using Equation (3)

$$R_i = \frac{x_i}{n} \tag{3}$$

The initial EWMA of each prefix A_0 is set to 1.0. The later values for each time interval T_i $(i = 1...n)$ is calculated using Equation (4).

$$A_i = \alpha.R_i + (1 - \alpha).A_{i-1} \tag{4}$$

where, α is the smoothing factor which lies in the open interval $(0, 1)$. This updated EWMA value is then used for making a caching decision in the next time interval.

For implementing the IBPC approach, a data structure Distinct Request Set (DRS) is created. DRS stores the content name ($name$), list of interfaces from which the requests for the content is received ($listOfInt$), and rating of the content ($rating$). Algorithm 3 describes how the DRS data structure gets updated on the arrival of an interest packet.

The modified NDN forwarding pipeline is shown in Figure 3. Whenever a router receives an interest packet, the router searches its name in the DRS. If the name is found, then the incoming interface is searched in $listOfInt$ of the matching entry. If it is not found, then the incoming interface is added to $listOfInt$. If the name is not found in the DRS, then a new entry is inserted in DRS having $name$ field equal to the name of the received interest packet, $listOfInt$ having the incoming interfaces on which the interest packet is received, and $rating$ field is set to 1.0. The rating of content gets updated using Algorithm 4. After every Tp period, the rating of the content is recalculated using Equation (4). For restricting the size of DRS, a parameter k is used that decides the size of DRS. If the DRS size is more than $k * CS - size$, then the entries with lower DRS are removed to make their size equal to $k * CS - size$,

Algorithm 3 OnInterest

INPUT: $interest, inFace$

```
1: it := DRS.search(interest.name)
2: if it ≠ NULL then
3:     drs := Create < Object > (DRS)
4:     drs.name := interest.name
5:     drs.listOfInt.insert(inFace)
6:     drs.rating := 1.0
7:     DRS.insert(drs)
8: else
9:     if it.listOfInt.serach(inFace) = NULL then
10:        it.listOfInt.insert(inFace)
11: Usual interest packet process
```

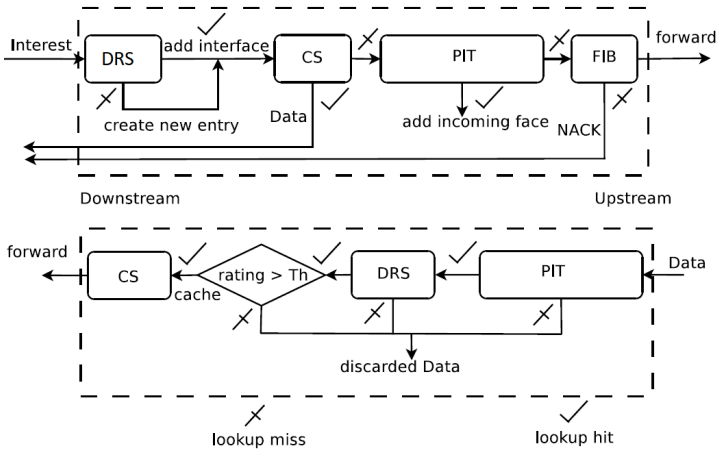


Fig. 3 Modified NDN Forwarding Pipeline

and the value of k is 20. The decision of whether to cache a content or not is taken using Algorithm 5.

Algorithm 4 PeriodicPopularityUpdation

INPUT:

```
1: for it = DRS.begin(); it ≠ DRS.end(); it++ do
2:     R = it.listOfInt.size()/n
3:     it.rating := α.R + (1 − α).it.rating
4:     it.listOfInt.clear()
5: truncate(DRS, k)
```

When a router receives a data packet, it searches for the corresponding entry in the PIT. If the entry is not found data packet is dropped. Otherwise,

Algorithm 5 OnData

INPUT: $interest, inFace$

```
it := DRS.search(data.name)
if it.rating > Th then
    Add data to CS
Usual data process
```

the router searches DRS for the matching entry. If the entry is found, then the *rating* corresponding to the entry is compared with a pre-existing threshold Th. If *rating* is greater than the Th, then the content is cached; else, it is dropped.

5.1 Mathematical Analysis

Before deploying the IBPC approach, the time period after which the popularity gets updated (Tp), the detection threshold (Th), and the alpha (α) should be known. Consider n number of contents $C_1, C_2, ..., C_n$ in the increasing order of their request rates $\lambda_1, \lambda_2, ..., \lambda_n$. Suppose content C_1 to C_{k-1} are popular contents and C_k to C_n are unpopular contents. The lower bound of the unpopular content's request rate will be the request rate of the most popular content among the unpopular content, i.e., λ_k. To ensure that no unpopular content is requested within the detection interval, the value of Tp should be less than the $1/\lambda_k$.

The value of Tp should be greater than $1/\lambda_{k-1}$ for ensuring that the least popular contents among the popular contents must be cached. Estimating the value of Tp in a real environment is time taking, but an approximate Tp can also give good accuracy for predicting popular content. Detection threshold Th decides the popularity threshold for the popular contents. It can vary depending upon the traffic and the number of interfaces from which the traffic is entering the router. Experimentally Th value from 0.4 to 0.6 found better. The value of alpha depends on the fluctuation of the traffic. It stabilizes the popularity of the content. Experimentally α value from 0.4 to 0.5 gives a better result.

5.2 Storage and Computational Overhead

IBPC approach stores a single entry for each content, which is its popularity in the DRS. The size of DRS is restricted by truncating its size to $CS - size * k$ after the detection interval Tp. If on an average n number of unique packets are received within the detection interval, then the Storage Overhead (SO) of DSR can be given by Equation (5).

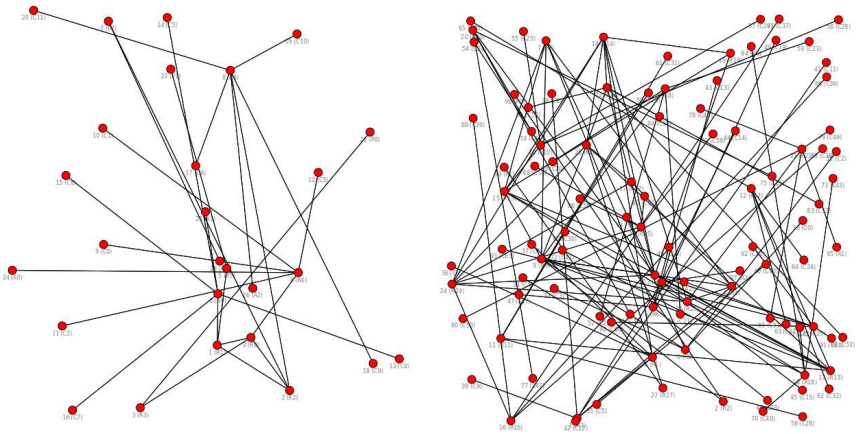
$$SO = O(CS - size * k + n) \tag{5}$$

When the router receives a content, then the counter corresponding to the content increases by 1. After Tp time interval the popularity of each content is updated using Equation (4). Thus, the computational overhead of the IBPC approach will depend on the rate of the content received by the router and Tp. Suppose, λ is the rate of arrival of content, then the computational Overhead (CO) of the IBPC approach can be given by Equation (6).

$$CO = \lambda + n/Tp \tag{6}$$

6 Experimental Result

The performance evaluation of the IBPC approach has been done using ns3 [3] based ndnSIM [2] simulator. The simulator is installed on a machine having Intel Core^{TM} i7 processor with 16GB RAM and Ubuntu 16.04 as the operating system. Two topologies, i.e., Xie-Complex (XC) topology, and the German Research Network (DFN) topology, are used to evaluate the IBPC approach. The evaluation of the proposed approach is divided into two phases- 1) Attack Modeling and 2) Countermeasure.



(a) XC Topology (b) DFN Topology

Fig. 4 Topologies used for the simulation

6.1 Attack Modeling

The XC topology consists of 12 consumers (C0-C11), 3 attackers (A0-A2), 3 publishers (P0-P2), and 9 routers (R0-R8). The DFN topology consists of 64 consumers (C0-C63), 4 attackers (A0-A3), 2 publishers (P0-P1), and

Table 2 Simulation parameters for both the topologies

Parameter	Value
CS Size	100, 150, 200, 250, 300
PIT Size	12000KB
Consumer's Request Frequency	100
Attacker's Request Frequency	500
Number of unique contents per prefix	1000
Fraction of unpopular contents used for performing attack	0.15
Simulation Time	200s

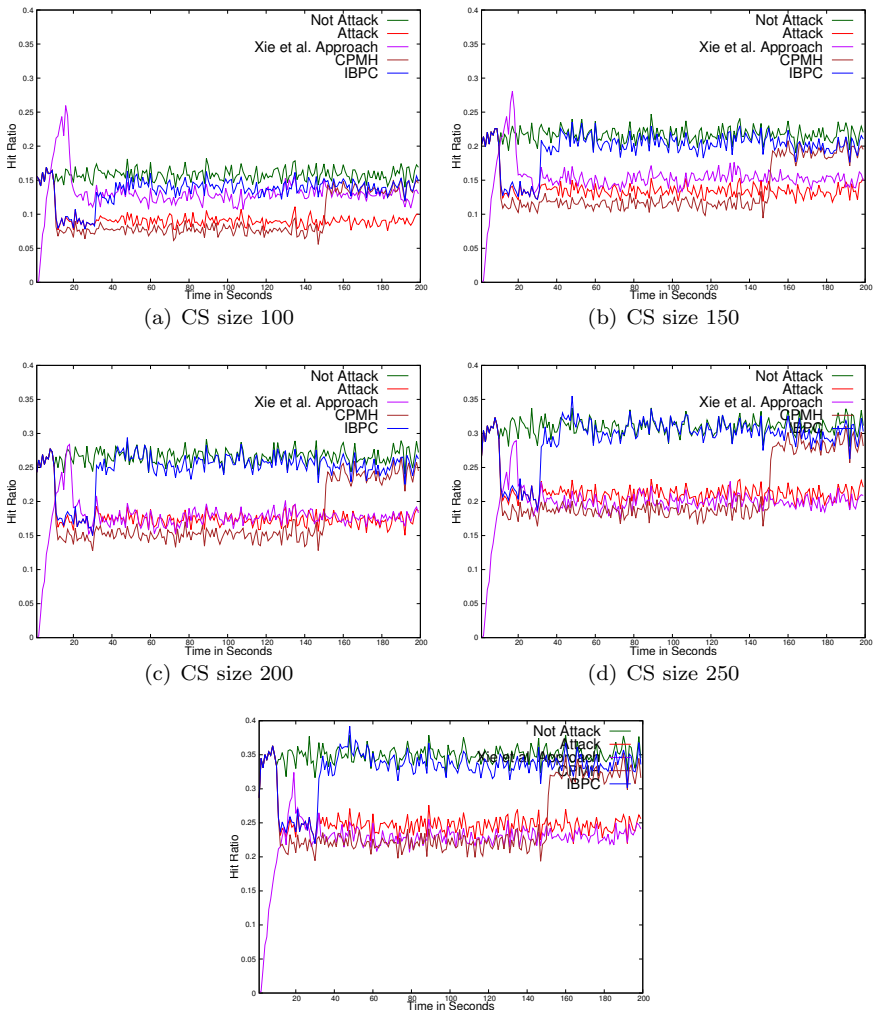
30 routers (R0-R29). The relative position of consumers, producers, attackers, and routers for XC and DFN topology is shown in Figure 4(a) and Figure 4(b) respectively. The simulation parameters, which are common for both the topologies, are shown in Table 2.

Different scenarios are simulated with changing CS size (100, 150, 200, 250, and 300 entries) for analyzing attack's effect. The consumers request packets with a fixed frequency of 100 interest packets per second for both the topologies. These packets are generated using Zipf-Mandelbrot distribution [19], having the value of q and s set to 0.7. The content is requested after a time period, which is chosen using the exponential random variable. The simulation runs for 200s of simulation time, and the consumer application starts at 10s. Each consumer generates requests for 1000 unique contents for each prefix with a frequency of 100. For performing the attack, 15 percent of unpopular contents are chosen from the tail of the Zipf. The attacker application starts at the 20s and requests packets with a frequency of 500. The effect of attack w.r.t time for XC topology and DFN topology is given in Figure 5 and Figure 6 respectively. The line graph having green colour shows the variation of hit ratio for the scenarios on which no attack is performed, whereas the line graph having red colour shows the scenarios on which the attack is performed. The decrement in the hit ratio can be seen for all the scenarios of both the topologies. Figure 7(a) and Figure 7(b) show the number of hits for XC topology and DFN topology for the attack (red colour) scenarios and the scenarios in which no attack (green colour) is performed.

6.2 Countermeasure

The IBPC approach is implemented using ns3 [3] based ndnSIM [2] network simulator. The value chosen for constants and thresholds used for simulating the IBPC approach has already been discussed in Section 6.1. The values of Tp, Th, and α are taken as 1s, 0.4, and 0.5, respectively, for both the topologies. For the evaluation of the IBPC approach, four scenarios have been considered:-

1. **Not Attack**: This is the scenario in which no attack is performed.



(a) CS size 100 (b) CS size 150

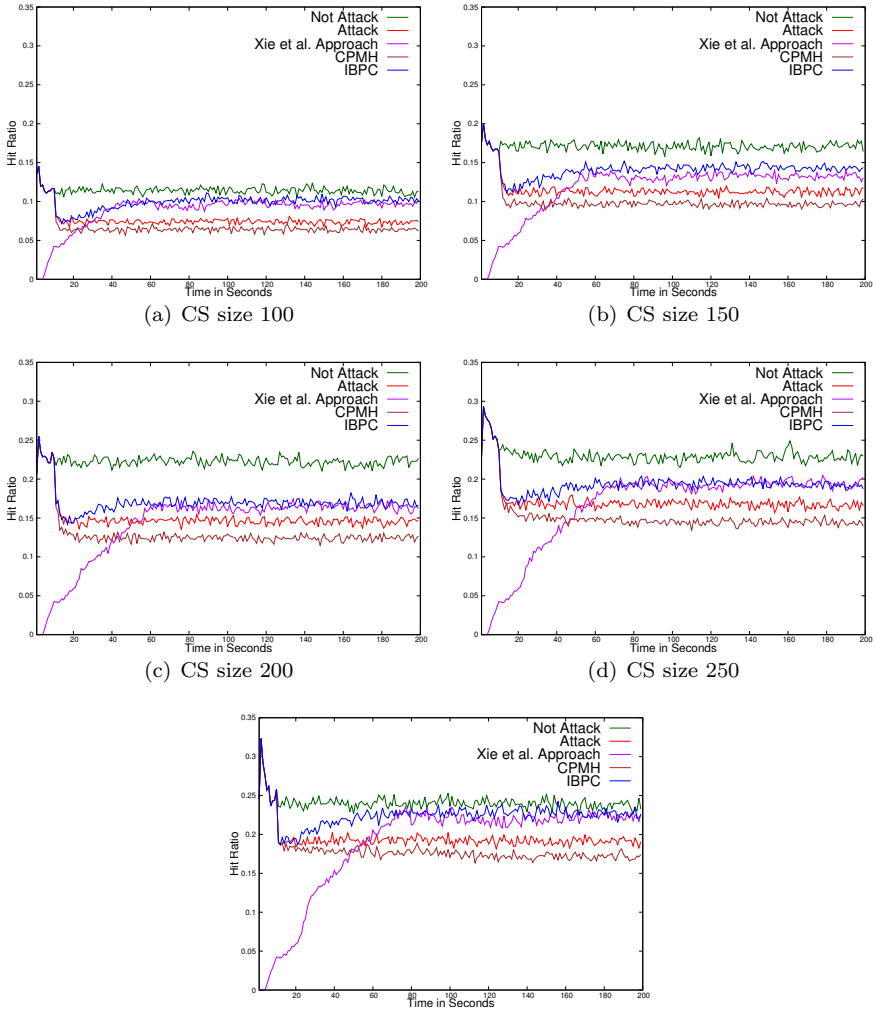
(c) CS size 200 (d) CS size 250

(e) CS size 300

Fig. 5 Number of hit ratio with respect to time for XC topology

2. **Attack**: This is the scenario in which the attackers are active, but no countermeasure is applied.
3. **Xie et al. Approach**: This is the scenario in which the attackers are active, and CPA mitigation approach proposed by Xie et al. [22] is applied.
4. **CPMH**: This is the scenario in which the attackers are active, and CPMH [12] CPA mitigation approach is applied.
5. **IBPC Approach**: This is the scenario in which the attackers are active, and the proposed IBPC mitigation approach is applied.

The value of p and q for the Xie et al. approach are 20 and 1, respectively, for both the topologies. Figure 5 shows that the hit ratio for the Xie et al.



(a) CS size 100 (b) CS size 150

(c) CS size 200 (d) CS size 250

(e) CS size 300

Fig. 6 Number of hit ratio with respect to time for DFN topology

approach increases exponentially at the beginning, then it decreases when the attackers become active in the 20s. The IBPC approach does not show any deviation during the attack. IBPC approach has a better hit ratio as compared to Xie et al. approach and CPMH for all the CS sizes. Also, it can be seen from Figure 7(a) that the number of hits for the IBPC approach is greater than Xie et al. approach and CPMH for all the CS sizes.

Figure 6 shows that in the case of DFN topology, Xie et al. approach and the CPMH fails to mitigate CPA. The IBPC approach has a greater hit ratio for all the scenarios. Also, it can be seen from Figure 7(b) that the number of hits for the IBPC approach is greater than the Xie et al. approach and CPMH

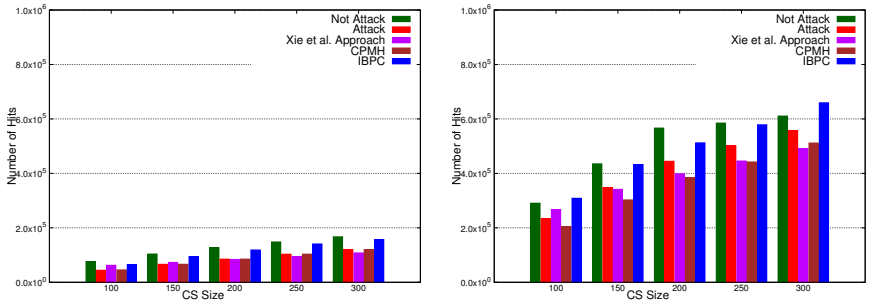


Fig. 7 Number of hits of the legitimate consumer for different CS size for (a) XC and (b) DFN topology

for all the CS sizes. The number of hits per second for Xie et al. approach increases exponentially at the beginning of the XC topology. After the start of the attack, the number of hits for Xie et al. approach decreases to 280. With the increase in time, this value fluctuates from 300 to 220. On the other hand, the IBPC approach is better than Xie et al. approach and CPMH in terms of the number of hits of legitimate consumers. The value of the number of hits for the IBPC approach is slightly less than the number of hits in the case of no attack scenario. Similar behavior can be seen in the case of DFN topology. In DFN topology, the IBPC approach gives greater hits as compared to no attack scenario.

The hit ratio for the IBPC approach is higher than the hit ratio for Xie et al. approach and CPMH for both the topologies, as illustrated through Figure 5 and Figure 6. The decrease in the popularity of unpopular content is shown in Figure 8. It can be seen that as simulation time increases from 0s to 200s, the average popularity rating of the unpopular contents decreases for both the topologies. The results and its discussion establish the fact that the proposed approach performs better than Xie et al. approach and CPMH in all the scenarios.

7 Conclusion

CPA is one of the severe attacks in NDN. The detection of CPA is very difficult as CPA does not follow any fixed pattern. Out of the two CPAs, i.e., LDA and FLA, FLA is more challenging to tackle. LDA can be countered by caching content based on its local popularity. However, the countermeasure, which is effective against LDA, fails to mitigate FLA. FLA manipulates local popularity; therefore, local popularity-based caching schemes can not mitigate FLA. The IBPC approach has successfully mitigated LDA and FLA through the use of interface-based popularity rating. Nonetheless, the storage and CPU overhead of the IBPC approach is comparatively low, proving its efficacy.

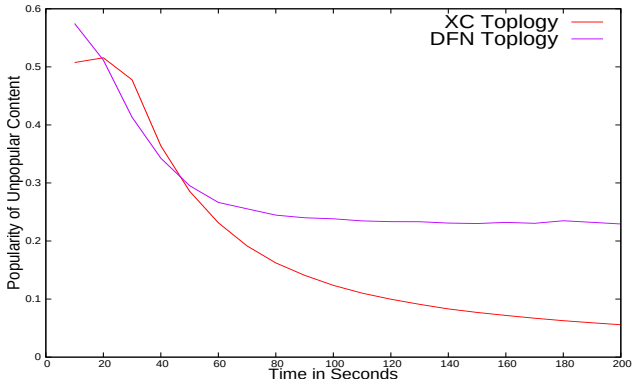


Fig. 8 Reduction of the popularity of unpopular content w.r.t. Time

Declarations

Funding: Not Applicable.
Conflicts of interest: None.
Availability of data and material: Not Applicable.
Code availability: Not Applicable.
Authors' contributions : All the authors have immensely contributed for the research work

References

1. Afanasyev, A., Mahadevan, P., Moiseenko, I., Uzun, E., Zhang, L.: Interest flooding attack and countermeasures in named data networking. In: 2013 IFIP Networking Conference, pp. 1–9. IEEE (2013)
2. Afanasyev, A., Moiseenko, I., Zhang, L., et al.: ndnsim: Ndn simulator for ns-3. University of California, Los Angeles, Tech. Rep **4** (2012)
3. Carneiro, G.: Ns-3: Network simulator 3. In: UTM Lab Meeting April, vol. 20, pp. 4–5 (2010)
4. Conti, M., Gasti, P., Teoli, M.: A lightweight mechanism for detection of cache pollution attacks in named data networking. Computer Networks **57**, 3178–3191 (2013)
5. Croarkin, C., Tobias, P., Zey, C., et al.: Single exponential smoothing (2002)
6. Deng, L., Gao, Y., Chen, Y., Kuzmanovic, A.: Pollution attacks and defenses for internet caching systems. Computer Networks **52**(5), 935–956 (2008)
7. Deng, L., Gao, Y., Chen, Y., Kuzmanovic, A.: Pollution attacks and defenses for internet caching systems. Computer Networks **52**(5), 935–956 (2008)
8. Gao, Y., Deng, L., Kuzmanovic, A., Chen, Y.: Internet cache pollution attacks and countermeasures. In: Proceedings of the 2006 IEEE International Conference on Network Protocols, pp. 54–64. IEEE (2006)
9. García, G., Beben, A., Ramón, F.J., Maeso, A., Psaras, I., Pavlou, G., Wang, N., Śliwiński, J., Spirou, S., Soursos, S., et al.: Comet: Content mediator architecture for content-aware networks. In: Future Network & Mobile Summit (FutureNetw), 2011, pp. 1–8. IEEE (2011)
10. Gilks, W.R., Richardson, S., Spiegelhalter, D.: Markov chain Monte Carlo in practice. CRC press (1995)
11. Jacobson, V., Mosko, M., Smetters, D., Garcia-Luna-Aceves, J.: Content-centric networking. Whitepaper, Palo Alto Research Center pp. 2–4 (2007)

12. Kamimoto, T., Mori, K., Umeda, S., Ohata, Y., Shigeno, H.: Cache protection method based on prefix hierarchy for content-oriented network. In: 2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC) (2016)

13. Karami, A., Guerrero-Zapata, M.: An anfis-based cache replacement method for mitigating cache pollution attacks in named data networking. Computer Networks **80**, 51–65 (2015)

14. Koponen, T., Chawla, M., Chun, B.G., Ermolinskiy, A., Kim, K.H., Shenker, S., Stoica, I.: A data-oriented (and beyond) network architecture. SIGCOMM Comput. Commun. Rev. **37**(4), 181–192 (2007)

15. Lauinger, T., Laoutaris, N., Rodriguez, P., Strufe, T., Biersack, E., Kirda, E.: Privacy implications of ubiquitous caching in named data networking architectures. Technical Report TR-iSecLab-0812-001, ISecLab, Tech. Rep. (2012)

16. Lauinger, T., Laoutaris, N., Rodriguez, P., Strufe, T., Biersack, E., Kirda, E.: Privacy risks in named data networking: What is the cost of performance? ACM SIGCOMM Computer Communication Review **42**(5), 54–57 (2012)

17. Park, H., Widjaja, I., Lee, H.: Detection of cache pollution attacks using randomness checks. In: Communications (ICC), 2012 IEEE International Conference on, pp. 1096–1100. IEEE (2012)

18. Salah, H., Alfatafta, M., SayedAhmed, S., Strufe, T.: Comon++: Preventing cache pollution in ndn efficiently and effectively. In: 2017 IEEE 42nd Conference on Local Computer Networks (LCN), pp. 43–51 (2017)

19. Silagadze, Z.: Citations and the zipf-mandelbrot's law. arXiv preprint physics/9901035 (1999)

20. Singh, V.P., Ujjwal, R.: Gini impurity based ndn cache pollution attack defence mechanism. Journal of Information and Optimization Sciences **41**(6), 1353–1363 (2020)

21. VNI, C.: Cisco visual networking index: Forecast and trends, 2017-2022 (2016)

22. Xie, M., Widjaja, I., Wang, H.: Enhancing cache robustness for content-centric networking. In: INFOCOM, 2012 Proceedings IEEE, pp. 2426–2434. IEEE (2012)

23. Xu, Z., Chen, B., Wang, N., Zhang, Y., Li, Z.: Elda: Towards efficient and lightweight detection of cache pollution attacks in ndn. In: Local Computer Networks (LCN), 2015 IEEE 40th Conference on, pp. 82–90. IEEE (2015)

24. Yao, L., Zeng, Y., Wang, X., Chen, A., Wu, G.: Detection and defense of cache pollution based on popularity prediction in named data networking. IEEE Transactions on Dependable and Secure Computing (2020)

25. Zhang, G., Liu, J., Chang, X., Chen, Z.: Combining popularity and locality to enhance in-network caching performance and mitigate pollution attacks in content-centric networking. IEEE Access **5**, 19012–19022 (2017)

Naveen Kumar has done his Bachelors of Technology (Computer Science and Engineering) from U.P. Technical University, Lucknow, in 2012. He has done his Masters of Technology (Software Engineering) in the Computer Science and Engineering Department at MNNIT Allahabad, Prayagraj, India in 2014. He is currently pursuing **Ph.D.** in the Department of Computer Science and Engineering at MNNIT Allahabad. His research interest includes peer to peer systems, future internet technologies, network security, and Named Data Networking.

Dr. Shashank Srivastava has done his Bachelors of Technology (Computer Science and Engineering) from U.P. Technical University, Lucknow. He has done **M.S.** in Information Security from Indian Institute of Information Technology Allahabad, India. He obtained his **Ph.D.** degree in Information Technology from Indian Institute of Information Technology Allahabad, India in 2014. He is currently working as Assistant Professor in the Department of CSE, MNNIT Allahabad, 211004, India. He possesses an experience of more than six years in the field of teaching and research. He is having the Membership of IEEE, ACM, CSI and CRSI (Cryptographic Research Society of India). He has supervised one doctoral thesis and currently guiding four research scholars at MNNIT Allahabad. His areas of expertise are Software Defined Networking (SDN), Named Data Networking (NDN), Network flow optimization and security, information security, and future internet technologies. He has published various research papers in reputed journals and conferences.