Received: July 6, 2020. Revised: September 28, 2020.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Efficient Secured PIT Management and Attack Detection Strategy Enhanced by CSOA-DCNN Algorithm in a Named Data Networking (NDN)

Ramachandira Moorthy Buvanesvari[1]* Kanagaraj Suresh Joseph[2]

[1]*Department of Computer Science and Engineering,*
Perunthalaivar Kamarajar Institute of Engineering and Technology, Karaikal, Puducherry, India
[2]*Department of Computer Science, Pondicherry University, Kalapet, Puducherry, India*
* Corresponding author's Email: tsbuvana@yahoo.in

Abstract: Named Data Networking (NDN) is a developing Internet design that utilizes a new network communication model dependent on the identity of Internet content. Its core component, the Pending Interest Table (PIT) serves an important role of recording Interest packet information. In managing PIT, the issue of flow PIT measuring has been very challenging because of the huge use of long Interest lifetime especially when there is no adaptable replacement strategy, subsequently affecting PIT performance. Named Data Networking (NDN) might experience some emerging threats such as Interest Flooding Attacks (IFA). In this paper, we focus on the IFA that can seriously devour the memory resource for the Pending Interest Table (PIT) of each included NDN router by flooding a huge amount of malicious Interests with spoofed names. To extricate the pressure of PIT attacked by IFA, we propose a methodology of efficient Secured PIT management and attack detection strategy by using a cuckoo search optimization algorithm-Deep convolutional neural network (CSOA-DCNN) algorithm in Named Data Network. The CSO algorithm initially utilizes a learning technique and afterward considers improved search operators and deep convolutional neural network architecture (DCNN) for classification. The network simulation tool is utilized to design and calculate PIT management. The results of the study on a 20 Gbps gateway trace shows that the corresponding PIT contains 1.5 M entries, and the lookup, insert and delete frequencies are 1.4 M/s, 0.9 M/s and 0.9 M/s. The contribution of this study is significant for Interest packet management in NDN routing and forwarding systems.

Keywords: Named data networking (NDN), Pending interest table (PIT), Interest flooding attacks (IFA), Cuckoo search optimization algorithm (CSOA), Deep convolutional neural network (DCNN), Interest satisfaction rate (ISR).

1. Introduction

Named Data Networking (NDN) is a network design which follows the content-centric approach, which focuses on the content itself instead of where the content is from [1]. As in other content-centric proposals, NDN packets have all inclusive one of kind names, with the goal that Data packets can be gotten, cached, and redistributed in the network [2]. In NDN, the forwarding process, the Pending Interest Table (PIT) is one of the important components [3, 4]. Showing up Interest packets are forwarded to the next-hop dependent on a Forwarding Information Base (FIB) [5] lookup only if the PIT finds no pending Interest packet with the same name. The PIT additionally stores the destination information for Data packets [6]. For every Data packet, the PIT is questioned to locate the incoming face(s) that mentioned the content, and afterward, the Data packet will be delivered and its content name is erased from the PIT [7]. Subsequently, the PIT requires per-packet update, including memory, writes. The PIT memory sizes expand as the link speed increases, which make space-limited fast memory devices infeasible to use [8, 9]. In each PIT section, the content name needs to be stored. The names are similar to URLs (Uniform Resource Locator), and today's URLs commonly require tens of bytes of storage [10]. For example, the URLs for the photos and recordings on well-known social networking websites, which include long hash numbers, are excess of 80 bytes long. Besides, some sites include article names in the URLs, making the URLs longer. Therefore, designing a fast and scalable Pending Interest Table is challenging [11].

We propose the network-wide solution to the adaptable Pending Interest Table issue by utilizing the cuckoo search optimization algorithm- Deep convolutional neural network (CSOA-DCNN) algorithm in Named Data Network. One of the recent nature-inspired metaheuristic algorithms is cuckoo search (CS) that was developed by Yang and Deb [12, 13]. A mix of these Levy flight advantages with nearby and worldwide search capacities makes CS as one of the most effective optimization algorithms. Low flexibility is the main drawback of this paper [14, 15]. In our structure, we organize network routers as either core routers or edge routers, similarly as in today's Internet. Subsequently, the core router PIT memory necessity is greatly diminished. However, a unique collision, a classical problem emerges [16, 17]. To ensure packet delivery, loosen the Interest total requirement in core routers.

The current pattern of Internet exercise through access speed required for data traversing, a tendency for the PIT to flood with resulting service disruption and possible network collapse is consistently practical [18]. Rapid Alubady et al., (2017) [19] has proposed a smart scheme named Adaptive Life Time (ALT) to deal with the Interest Lifetime at content routers that will adapt Lifetimes as a function of system load. With such a plan, it would break down the speculation that expresses that intermediate nodes do not manage Lifetime values. The main drawback is less security.

Tan Nguyen et al., (2019) [20] has considered the instance of Denial of Service. Although the Interest Flooding Attack (IFA) has been great extent studied and mitigated through NACK (Negative Acknowledgement) packets in pure NDN networks, they demonstrated in this paper through exploratory appraisals that there are still few different ways to mount such an attack, and particularly with regards of coupling NDN with IP (Internet Protocol), that can scarcely be tended by current arrangements. At last, considered a genuine deployment scenario where NDN is combined with IP to carry HTTP (Hypertext Transfer Protocol) traffic. The process takes more time. There is some delay. By considering information from the test bed, show the productivity of overall discovery technique.

Ahmed et al., (2020) [21] has introduced a strategy called MSIDN (Mitigation of Sophisticated Interest flooding-based DDoS attacks in Named Data Networking) expects to moderate attackers at the source without influencing legitimate users. MSIDN depends on data producers' feedback to start attack mitigation.

2. Proposed methodology

2.1 Overview of NDN architecture

In this section, we briefly describe the NDN architecture and its fundamental ideas [22]. Content sources promote/publish their accessible content items in the network by giving the Publication packets, which typically incorporate a prefix-based content name or some other form of content identifier (ID) [23]. Each switch, after receiving such packets, records the incoming interface and the content name in its Forwarding Information Base (FIB). ICN (Information-Centric Networking) also natively supports quality-of-service and content caching [24]. The latter methodology changes caches into alternative content sources and empowers joint optimization of sending and caching functions. These packets are then sent through an arrangement of routers towards the content source or cache, as indicated to FIB passages for mentioned content. In case those different sources are holding similar content, some kinds of mediated topology management function could be utilized to choose the best source or even to empower multi-chunk content delivery.

The data packet is sent by the reverse path towards the user is illustrated in Fig. 1. This is called pull-based communication model and it guarantees that the user gets just explicitly mentioned content. In order, for the routers to be able to deliver Data packets to the users, every router is furnished with PIT. The latter contains entries for all "not yet satisfied" Interest packets and their approaching interface.

2.1.1. NDN names

NDN names are application-dependent and opaque to the network, yet they all offer the basic

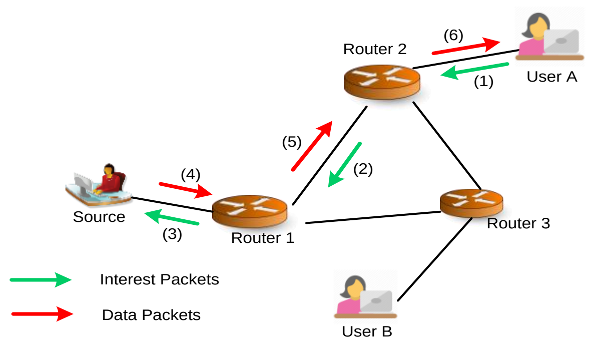

Figure. 1 Basic NDN communication model

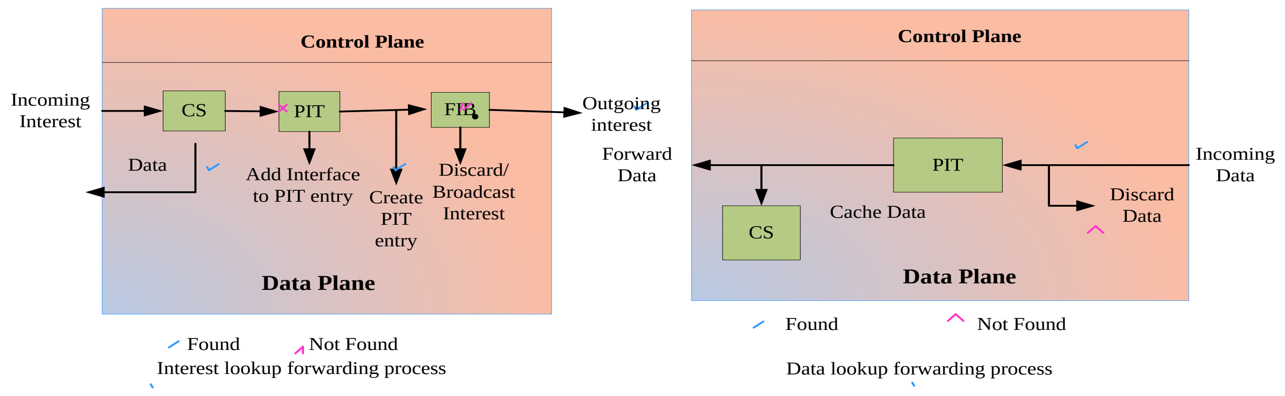

Figure. 2 Packet lookup and forwarding process

attributes – hierarchically organized and composed of explicitly delimited components. A typical example of NDN name is reversed domain names followed by directory-style path, e.g., org/ieeeinfocom/2013/cfp.html.

2.1.2. Requester-driven communication model

There are two sorts of packets in NDN – Interest packet and Data packet. Generally, Interest packet is the request and The Data packet is the response. An Interest packet conveys a name, or an identifier, that specifies the desired data; the name is also encapsulated in Data packet, demonstrating the content of it. NDN receives request-driven communication model, in which a request sends out an Interest packet for desired content, and a server restores the content inside a Data packet.

2.1.3. Packet forwarding

An NDN router has different forwarding processes for Interest packet and Data packet, which are illustrated by Fig. 2 (a) and Fig. 2 (b) respectively. From the figures we can see that NDN router keeps three tables: Forwarding Information Base (FIB), Pending Interest Table (PIT) and Content Store (CS). FIB is the NDN routing table; PIT, as the name proposes, keeps track of un-satisfied Interest packets, as well as their approaching interfaces; and CS deliberately caches Data packets to serve subsequent Interest packets mentioning the same content.

2.2 Interest flooding attack

Malicious/compromised users may exploit the PIT-based forwarding mechanism of NDN to dispatch the IFA, which is considered as one of the most genuine sorts of DDoS attacks on NDN. Every router, after getting each of these packets, will make a section in its PIT and will forward the packet to next-hop node. As indicated by NDN rules, an entry is expelled from PIT in accompanying two cases:

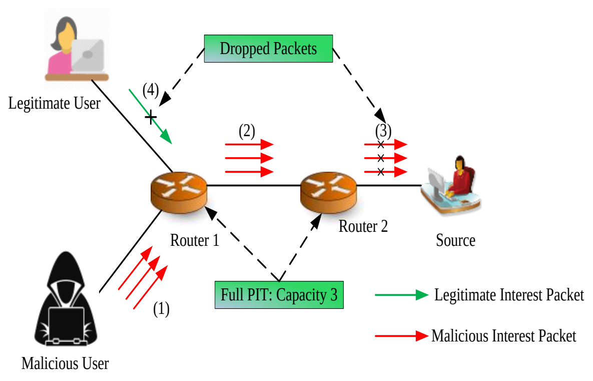

Figure. 3 Interest flooding attack on NDN

- Entry expired
- Router received the corresponding Data packet before the entry expiration.

As indicated by above, the best attacking methodology is giving Interest packets for non-existent content. In this situation, the bogus entries will remain in PIT however much as expected. The objective of assailant is to rapidly fill in PIT and to keep it full, so that the Interest packets originated from legitimate users will eventually be dropped.

In Fig. 3, illustrate a simple example of IFA in NDN. Assume that PIT limit in every router is 3 entries. The attacker's methodology is to send 3 bogus Interest packets for non-existent content. These packets will fill in the PITs of two routers. The source will drop these packets, since they demand non-existent content. In any case, the comparing passages will remain in the PITs until they lapse. After the lapse, the attacker will issue 3 new Interest packets, aiming at keeping the PITs every case full. This way, some, or even all, Interest packets of legitimate users will be dropped.

2.3 Pending interest table

This section discussed the differences between edge and core routers and then features its prerequisites. Edge routers connect consumers to ISP

Received: July 6, 2020. Revised: September 28, 2020.

Table 1. Pending interest table requirements

Line Rates	**Edge (1 G)**	**Core(10 G)**	**Core(100 G)**
Interfaces	thousands	hundreds	Tens
Best Case	0.156Mpps	1.563Mpps	15.625Mpps
Worst Case	1.25Mpps	12.5Mpps	125Mpps
Best Memory	0.65MiB	6.25MiB	62.5MiB
Worst Memory	5Mib	50MiB	500MiB

networks, and along these lines, the numbers of interfaces on edge routers are commonly large, arriving 64 thousand. The throughput isn't high for edge routers. Network traffic totals at edge routers and afterward enters the backbone networks. Core routers are sent in backbone networks, where the throughput rather than the quantity of interfaces is essential concern. A high-end router may contain different line cards, and hence its bandwidth can reach 10 Gbps, 100 Gbps, or more. For this situation, the quantity of entries in PIT is enormous, and PIT needs to be refreshed efficiently. Generally, it cannot consist many features on core routers as on edge routers, and the number of faces is small, extending from a couple of interfaces to tens of interfaces. The quantity of packets arriving at each face is additionally influenced by packet sizes. Since NDN can run on top of Ethernet straight forwardly, the packet size could be as little from 64 bytes, or as extensive from 1500 bytes. By and large, Interest packets are generally small, and Data packets are enormous. In a stream balance mode, one Data packet is moved for one Interest packet. The best case can be expected as 100-byte Interest packets and 1500-byte Data packets. In the worst case, the Data packets can be as little as Interest packets; therefore set 100 bytes for both of them.

Table 1 records both operation frequency and memory necessities, expecting the round trip time Trtt = 80 ms, SI = 100 Bytes, the best case SD = 1500 Bytes, and the worst scenario SD = 100 Bytes. We are not considering the situation where a large portion of Interest packets cannot be satisfied and have to wait for expiration, because this behavior could happen only when the PIT is under a flooding attack. From Table 1, for low-end routers, the memory size in worst case can be easily fit into SRAM (Static Random Access Memory). In any event, for the 10 Gbps case, it is possible to fit them into RLDRAM (Reduced Latency Dynamic Random Access Memory). However, the 100 Gbps case, its memory size cannot be fit into RLDRAM, in this manner DRAM (Dynamic Random Access Memory) has to

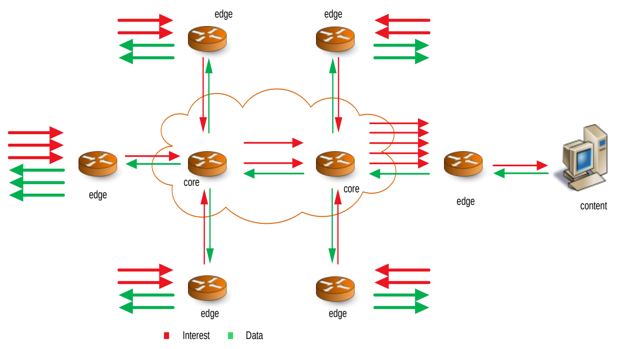

Figure. 4 System design

be utilized. It is significant that designing the PIT for worst case, since believe the PIT should be capable of handling the worst case traffic.

Fig. 4 shows the system design. In figure, Interest packets are amassed at edge routers and then enter the core network. The core routers basically forward all received Interest packets. The content supplier gets just one Interest request. Then one Data packet is replied and dispersed to the users. The whole packet handling methodology is straightforward to the users and content providers. The PITs in edge switches operate as described in the NDN design, where name strings are put away, and Interest aggregation is upheld.

2.4 CSOA-DCNN algorithm in named data network

The deep convolutional neural network (CNN) architecture proposed in this study is involved 3 convolutional layers interleaved with 2 pooling tasks, trailed by 2 completely associated layers. Like recently proposed feature learning approaches applied to environmental sound classification, the input to the network comprises of time-frequency patches taken from log-scaled melspectrogram representation of audio signal. In particular, here utilize to separate log-scaled mel-spectrograms with 128 components covering the dissemble frequency range (0-22050 Hz), utilizing a window size of 23 ms and a hop size of similar duration. Since the portions in our assessment the dataset are varying duration, we fix the size of input TF-patch to 3 seconds, i.e. TF patches are extracted arbitrarily from full log-Mel spectrogram of each audio during training as portrayed further down.

Given our input Y, the network is qualified to learn the parameters $\Phi$of composite nonlinear function $G(\cdot\,|\Phi)$which maps Y to output (prediction) W:

$$W = G\left(Y\middle|\Theta\right) = g_1(\dots.g_2(g_1(Y|\theta_1)|\theta_2)|\theta_J) \quad (1)$$

Where, each process $g_j(\cdot|\theta_j)$ is referred to as layer of network, with J = 5 layers in our proposed architecture. The first three layers $j \in \{1,2,3\}$ are convolutional, expressed as:

$$W_j = g(Y_j|\theta) = b(Z \times Y_j + a), \theta_l = [Z, a] \quad (2)$$

Where,$Y_j$as 3-dimensional input tensor consisting of N feature maps, Z is a collection of M 3-dimensional kernels, × represents a valid convolution, a represents vector bias term, and $b(\cdot)$ is a point-wise activation function. Thus, the shapes of Y_j, Z and W_j are M, d_0, d_1 , P, M, p_0, p_1 and $P, d_0 - m_{0+1}, d_1 - m_1 + 1$ respectively. Note that for first layer of our network $d_0 = d_1 = 128$, i.e., dimensions of input TF-patch. We apply strided max-pooling after the first two convolutional layers $l \in \{1,2\}$ using a stride size equal to pooling dimensions (provided below), which decreases the dimensions of the output feature maps and consequently speed up training and builds some scale invariance into the network. The final two layers $l \in \{4,5\}$ are fully-connected and consist of matrix product rather than convolution:

$$W_l = g(Y_l|\theta_l) = b(ZY_l + a)\theta_l = [Z, a] \quad (3)$$

Where, Y_l is flattened to a column vector of length M, Z has shape (P, N)b is a vector of length P and $b(\cdot)$ is a point-wise activation function. An authentication set is used to identify the parameter setting (epoch) achieve the highest classification accuracy, where prediction is performed by slicing the test sample into overlapping TF-patches (1-frame hop), making a prediction for each TF-patch and lastly choose the sample level prediction as class with highest mean output activation over all frames. The CSO algorithm is influenced by breeding technique of some cuckoo species related to Levy flight behavior of certain birds. At first some cuckoo birds lay their eggs in host bird nests. In any case, host birds might come to know, that these eggs are not its own and may either slaughter them or leave them. For simplicity in portraying the CSO, three conceptual rules are:

- A set of solution is represented by each cuckoo laying egg and select random nest for dump in it.
- A fraction of nest with highest quality fitness egg or best solution will be carried out to the next iteration.
- The availability of host nests is set by some value and the foreign egg is searched by the host bird with a probability $x_a \varepsilon(0-1)$ and to circumvent computational burden, this probability index is set by value $x_a = 0.25$.

For a cuckoo i, the new cuckoos (solutions) y_i^{u+1} generated by using Levy flight as:

$$y_i^{(u+1)} = y_i^u + \alpha \oplus Levy(\lambda) \quad (4)$$

Where, $\alpha > 0$ is the step size depends upon the type of problem of interest. Most time, the step size $\alpha = 1$ is selected. The product $\oplus$ means entry wise multiplication. The Levy flight is fundamentally a random walk in which the random step length is estimated from Levy distribution for large steps as:

$$Levy \propto t = u^{-\lambda}, (1<\lambda \leq 3) \quad (5)$$

The Eq. (2) has an infinite variance with an infinite mean. Here the consecutive jumps or steps of a cuckoo basically form a random walk process which obeys the power law step-length distribution with a heavy tail. Details of CSO algorithm is illustrated in Table given below.

Algorithm 1
Step 1: Initialize population 'P' with 'M' host randomly
Step 2: Evaluate fitness value of initial solution
Step 3: while i<MaxGen do
Step 4: Cuckoo via Levy flight is computed based on Eq. (5)
Step 5: Calculate fitness for cuckoo i
Step 6: Choose nest among M_j randomly
Step 7: If g_i is superior than g_j then
Step 8: Substitute j by attained solution
Step 9: end if
Step 10: Fraction of worse nests are neglected and constructed
Step 11: Adjust global best solution
Step 12:t = t + 1; 13: end while

3. Experimental results

In this section, we perform a simulation study to evaluate the PIT management and attack detection Strategy enhanced by CSOA-DCNN algorithm in named data network. We have implemented in ndnSIM platform. In this section, we first introduce our main simulation settings and then discuss the simulation results of two scenarios.

Received: July 6, 2020. Revised: September 28, 2020.

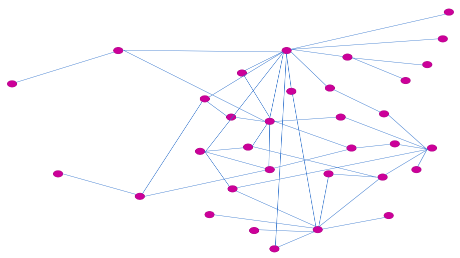
Figure. 5 Network topology used in simulation

Table 2. Simulation settings

Parameters Description	Values
PIT size	200
PIT entry life time (seconds)	4
Legitimate request rate	50
Malicious request	(10, 0.5, 5)
Simulation time (seconds)	0 - 500
Legitimate request time (seconds)	0 – 500
Malicious request time (seconds)	250 - 500

3.1 Experimental setup

The network topology that we utilized in our evaluation is represented in Fig. 5. The settings used in the simulation process are detailed in Table 2.

3.2 Evaluation results

3.2.1. Memory usage

From two Name Sets, we first measure their memory consumption of: 1) original size, i.e., directly stores the names as character strings in table, 2) NPT, 3) ENPT (S2TA) + hash table. The overall results are appeared in Table 3; more detailed results are given in Fig. 6 and Fig. 7. A few facts can be derived from these measurements. In addition, besides the name components, each NPT node stores extra information, such as state number, pointer to children, pointer to parent, etc.

Table 3. Name component statistics of two name sets

URL set	10M name set	8M name set
No. of names	8,844,747	7,645,481
No. of total components	24,912,123	26,235,596
Average component length (Byte)	6.15	14.35
Average no of components per name	2.43	14.35
Original size (MB)	182.35	413.87
No of components/ edges in NPT	12,339,081	4,680,563
NPT size (MB)	247.68	136.14
ENPT size (MB)	47.82	19.37
ENPT+ Hash Table size (M)	127.13	52.72
Compression ratio	64.44%	13.53%

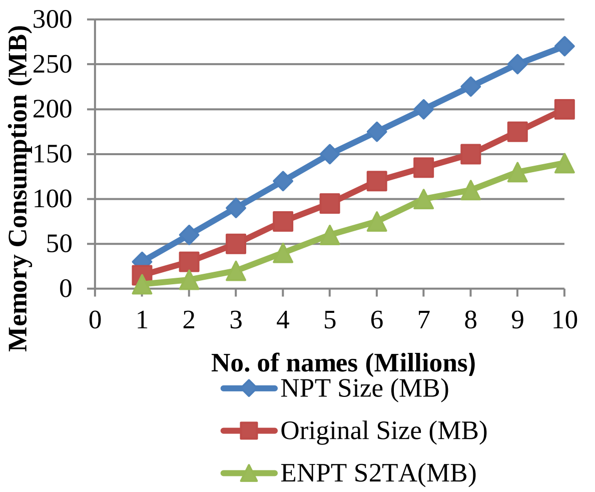

Figure. 6 10M name set memory consumption –original size, NPT size, ENPT size

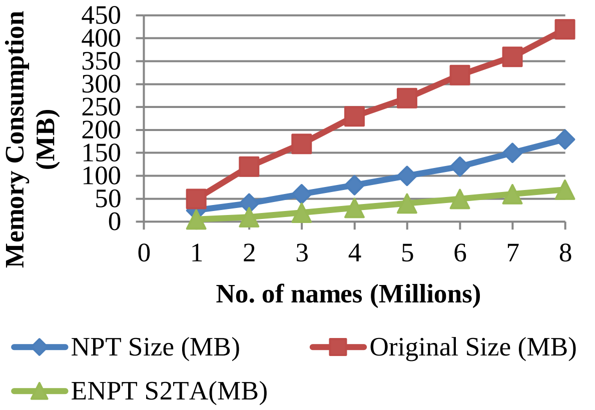

Figure. 7 8M name set memory consumption –original size, NPT size, ENPT size

Consequently, the NPT of 10M Name Set is considerably bigger than its original size. But this extra data makes NPT (Network Parameter Table) simpler to manage than storing names directly. Thus a hash table is required, for 10M Name Set, size of hash table is roughly 66.10 MB, and around 34.56 MB for 8M Name Set. Finally, compression ratio of 10M Name Set and 8M Name Set is 64.67% and 13.66%, respectively. Please keep in mind that, ENPT is logical data structure and we don't execute it straightforwardly. However, it is actualized by S2TA. We can consider of procedure on ENPT, yet actually operate the S2TA to carry out those tasks.

In fig. 8, we have compared the memory consumption of proposed algorithm with existing algorithms. The existing algorithms such as, MSIDN, HS and GLRT. We have compared the original size,

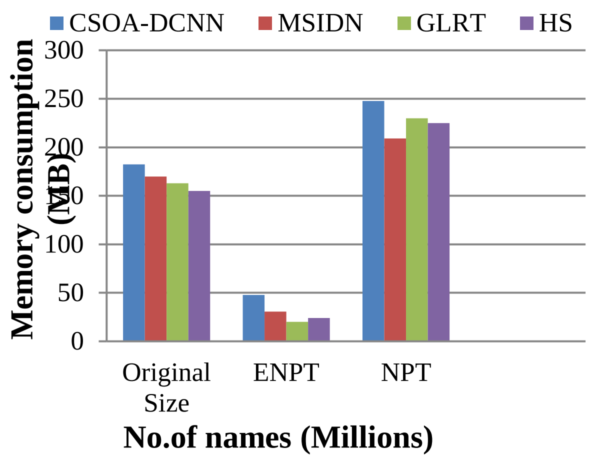

Figure. 8 Memory consumption comparison

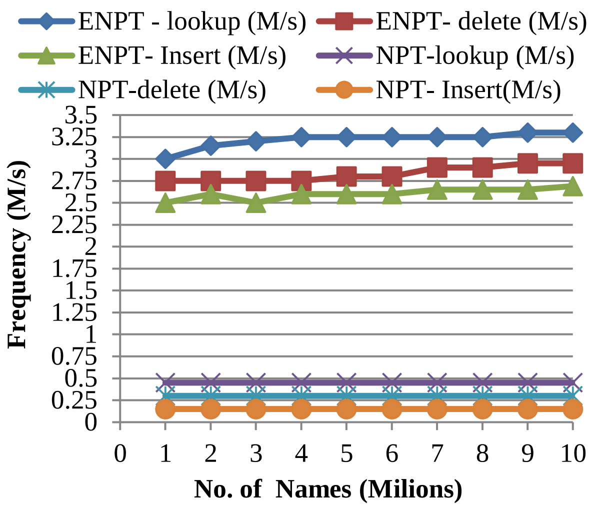

Figure. 9 Lookup, insert and delete performance for the 10M Name set

ENPT and NPT of proposed and existing algorithms. In this comparison the proposed method is better than existing methods.

3.2.2. Lookup, insert and delete performance

After these lines, measure the PIT access (lookup, insert and delete) performance by ENPT. Do not measure the update performance, because updating PIT means appending interfaces to specific PIT entry, whose presentation is extremely near to that of lookup. Deleting a name includes back following away, which is simple to implement by NPT since NPT has parent pointers. The deleting procedure in ENPT is: first go directly to leaf node, delete the leaf, then backtrack to see if the parent node still needs to be removed. If a parent node's all the children are deleted, and it does not have a pointer to PIT entry, it should be removed too. Because there is no such pointer to parent node in S2TA, we monitor the track of node information along the path to specific leaf when deleting a name. We do not delete the node and free space of ENPT deleting a node means the code of its preceding edge/component is liberated, and that code can be reused inside its CAS, just as corresponding space in S2TA. The lookup, insert and delete performance for two Name Sets are represented in Fig. 10 and Fig. 11, respectively. The lookup, insert and delete execution on 10M Name Set can achieve 3.26 M/s, 2.94 M/s and 2.68 M/s, respectively, and that on 8M Name Set accomplish 2.52 M/s, 1.82 M/s and 2.17 M/s, respectively. PIT gets frequency dependent on ENPT is extraordinarily advanced contrasted with that of NPT, which is more clearly depicted by speedups in Fig. 13 then backtrack to check whether the parent node still needs to be removed, and so forth. If a parent node's all the children are deleted, and it does not have a pointer to PIT entry, it should be removed too.

Deleting a node means the code of its preceding edge/component is freed, and that code can be reused inside its CAS, just as corresponding space in S2TA. 1 2 3 4 5 6 7 8 9 10 0.00 0.24 0.51 0.74 1.01 1.24 1.51 1.74 2.01 2.24 2.51 2.763.01 3.24 3.51 Frequency (M/s) of names (Million) ENPT-lookup (M/s) ENPT-delete (M/s) ENPT-insert (M/s) NPT-lookup (M/s) NPT-delete (M/s) NPT-insert (M/s) Fig. 9: Lookup, insert and delete performance for the 10M Name Set. 1 2 3 4 5 6 7 8 0.00 0.23 0.51 0.74 1.01 1.24 1.51 1.74 2.01 2.24 2.51 2.74 Frequency (M/s) of names (Million) ENPT-lookup (M/s) ENPT-delete (M/s) ENPT-insert (M/s) NPT-lookup (M/s) NPT-delete (M/s) NPT-insert (M/s) Fig. 10: Lookup, insert and delete execution for the 8M Name Set. 1 2 3 4 5 6 7 8 9 10 -4 -2 0 2 4 6 8 10 11 13 16 17 20 23 25 Speedup of names (Million) 8M Name Set - lookup speedup 8M Name Set - delete speedup 8M Name Set - insert speedup 10M Name Set - lookup speedup 10M Name Set - delete speedup 10M Name Set - insert speedup Fig. 10:

The lookup, insert and delete execution on 10M Name Set can achieve 3.26 M/s, 2.94 M/s and 2.68 M/s, respectively, and that on 8M Name Set accomplish 2.52 M/s, 1.82 M/s and 2.17 M/s, respectively. PIT gets frequency dependent on ENPT is extraordinarily advanced contrasted with that of NPT, which is more clearly depicted by speedups in Fig. 13 then backtrack to check whether the parent node still needs to be removed, and so forth. If a parent node's all the children are deleted, and it does not have a pointer to PIT entry, it should be removed too.

Deleting a node means the code of its preceding edge/component is freed, and that code can be reused inside its CAS, just as corresponding space in S2TA. 1 2 3 4 5 6 7 8 9 10 0.00 0.24 0.51 0.74 1.01 1.24 1.51 1.74 2.01 2.24 2.51 2.763.01 3.24 3.51 Frequency (M/s) of names (Million) ENPT-lookup (M/s) ENPT-delete (M/s) ENPT-insert (M/s) NPT-lookup (M/s) NPT-delete (M/s) NPT-insert (M/s) Fig. 9: Lookup, insert and delete performance for the 10M Name Set. 1 2 3 4 5 6 7 8 0.00 0.23 0.51 0.74 1.01 1.24 1.51 1.74 2.01 2.24 2.51 2.74 Frequency (M/s) of names (Million) ENPT-lookup (M/s) ENPT-delete (M/s) ENPT-insert (M/s) NPT-lookup (M/s) NPT-delete (M/s) NPT-insert (M/s) Fig. 10: Lookup, insert and delete execution for the 8M Name Set. 1 2 3 4 5 6 7 8 9 10 -4 -2 0 2 4 6 8 10 11 13 16 17 20 23 25 Speedup of names (Million) 8M Name Set - lookup speedup 8M Name Set - delete speedup 8M Name Set - insert speedup 10M Name Set - lookup speedup 10M Name Set - delete speedup 10M Name Set - insert speedup Fig. 10: PIT access frequency speedup based on ENPT. The lookup, insert and delete performance for two Name Sets are illustrated in Fig. 8 and Fig. 10, respectively. Obviously, PIT access frequency dependent on ENPT is phenomenally promoted compared with that of NPT, which is more clearly portrayed by speedups in Fig. 10.

In this Fig. 11 we are comparing lookup, insert and delete performance of ENPT for proposed algorithm with an existing algorithm. The lookup of proposed algorithm is 11.66% better than MSIDN, 38.88% better than GLRT and 34% better than HS. The insert of CSOA-DCNN is 7.84% better than MSIDN, 3.63% better than GLRT and 5.76% better than HS. The delete performance of proposed algorithm is 30% better than MSIDN, 8.3% better than GLRT and 9.09% better than HS algorithm. The proposed algorithm provides better performance than existing algorithms.

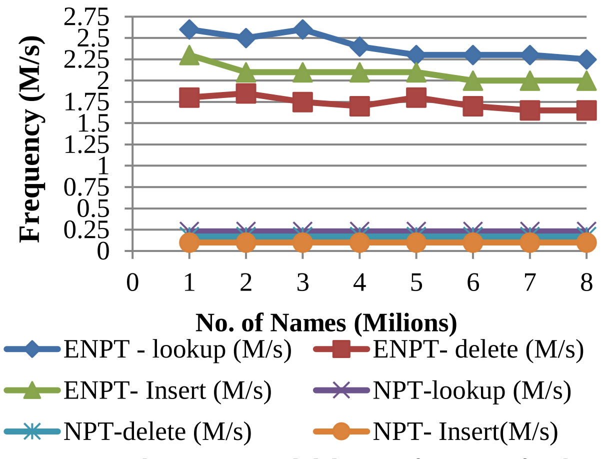

Figure. 10 Lookup, insert and delete performance for the 8M Name set

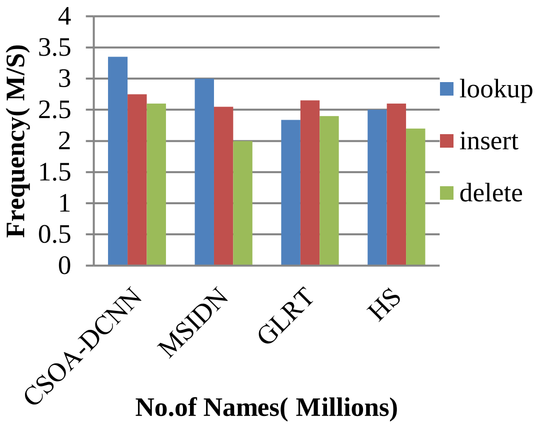

Figure. 11 Lookup, delete and insert performance of ENPT

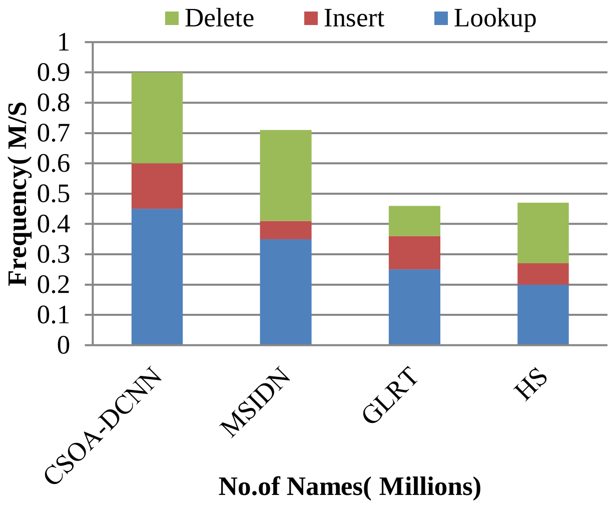

Figure. 12 Lookup, delete and insert performance of NPT

Fig. 12 shows the performance comparison of proposed algorithm with an existing algorithm. Here, we are comparing insert, delete and look up performance of NPT. The lookup of proposed algorithm 28.57% better than MSIDN algorithm, 80% better than GLRT and 40% better than HS. The insert of CSOA-DCNN is 60% better than MSDIN algorithm, 36.36% better than GLRT and 68% better than HS algorithm. The delete of proposed algorithm is same as MSIDN, 20% better than GLRT and 46% better than HS algorithm. From the comparison the proposed algorithm gives better result than the existing algorithms.

Lookup, Insert, Delete performance of various algorithms is shown in Fig. 11 and 12. The proposed algorithm is compared with three existing algorithms [13, 20, 21]. The proposed algorithm gives better result than existing three algorithms.

Received: July 6, 2020. Revised: September 28, 2020.

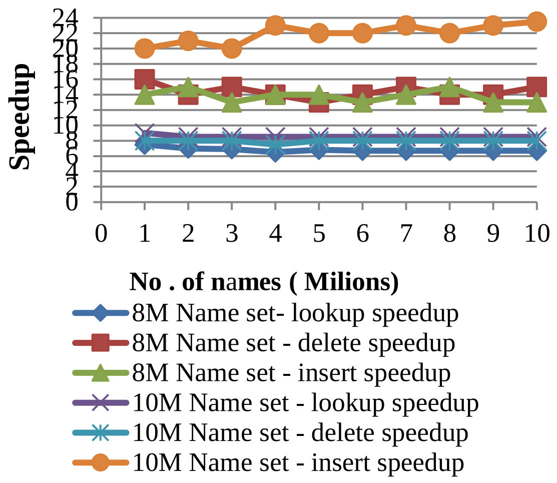

Figure. 13 PIT access frequency speedup based on ENP

4. Conclusion

In this paper we carry out, to the best of our knowledge, the first analytical examination on PIT dynamics, determining normal and maximum stationary values. We also construct a test high-speed platform permitting us to assess the accuracy of our model in the presence of synthetic traffic workload and trace-driven packet delay circulation. Concerns about the PIT state explosion communicated by past works on the basis of approximate calculations are disproved by this study. In this paper addressed three problems related with the PIT: 1) the size and access (lookup, insert, delete) frequency of PIT; 2) how to address the enormous size and high access frequency problem with a scalable solution; 3) where does PIT dwell inside a router. We copy NDN's application-layer working paradigms by transferring the existing IP applications to the NDN stage. In terms of lookup, insert and delete the performance of ENPT for proposed method is 11.66% better than MSIDN, 38.88% better than GLRT and 34% better than HS. The insert of CSOA-DCNN is 7.84% better than MSIDN, 3.63% better than GLRT and 5.76% better than HS. The delete performance of proposed algorithm is 30% better than MSIDN, 8.3% better than GLRT and 9.09% better than HS algorithm. The performance of NPT for proposed algorithm 28.57% better than MSIDN, 80% better than GLRT and 40% better than HS. The insert of CSOA-DCNN is 60% better than MSDIN, 36.36% better than GLRT and 68% better than HS algorithm. The delete of proposed algorithm is same as MSIDN, 20% better than GLRT and 46% better than HS algorithm. We evaluate the size and access frequency of PIT, which requests an efficient and scalable solution. Further research is required to design and investigate such PIT management.

References

[1] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang, "Congestion control in named data networking – A survey", *Computer Communications*, Vol. 86, No. 7, pp. 1-11, 2016. Available: 10.1016/j.comcom.2016.04.017.

[2] T. Nguyen H.L. Mai, G. Doyen, R. Cogranne, W. Mallouli, E. M. De Ocaand, and O. Festor, "A Security Monitoring Plane for Named Data Networking Deployment", *IEEE Communications Magazine*, Vol. 56, No. 11, pp. 88-94, 2018. Available: 10.1109/mcom.2018.1701135.

[3] H. Khelifi, S. Luo, B. Nour, H. Moungla, Y. Faheem, R. Hussain, and A. Ksentini, "Named Data Networking in Vehicular Ad Hoc Networks: State-of-the-Art and Challenges", *IEEE Communications Surveys & Tutorials*, Vol. 22, No. 1, pp. 320-351, 2020. Available:10.1109/comst.2019.2894816.

[4] R. Ullah, M. Rehman, and B. Kim, "Design and Implementation of an Open Source Framework and Prototype for Named Data Networking-Based Edge Cloud Computing System", *IEEE Access*, Vol. 7, No. 4, pp. 57741-57759, 2019. Available: 10.1109/access.2019.2914067.

[5] T. Yang, J. Li, C. Zhao, G. Xie, and X. Li, "Mathematical analysis on forwarding information base compression", *CCF Transactions on Networking*, Vol. 1, No. 1-4, pp. 16-27, 2018. Available: 10.1007/s42045-018-0010-1.

[6] V. Sivaraman, D. Guha and B. Sikdar, "Optimal Pending Interest Table Size for ICN With Mobile Producers", *IEEE/ACM Transactions on Networking*, Vol. 28, pp. 1-14, 2020. Available: 10.1109/tnet.2020.2988713.

[7] D. Gupta, S. Rani, S. Ahmed, and R. Hussain, "Towards Pending Interest Table Management Solutions in Named Data Networking", *Journal of Computational and Theoretical Nanoscience*, Vol. 16, No. 10, pp. 4271-4279, 2019. Available: 10.1166/jctn.2019.8512.

[8] R. Shubbar and M. Ahmadi, "A Filter-Based Design of Pending Interest Table in Named Data Networking", *Journal of Network and Systems Management*, Vol. 27, No. 4, pp. 998-1019, 2019. Available: 10.1007/s10922-019-09495-y.

[9] M. Majeed, S. Ahmed, and M. Dailey, "Enabling Push-Based Critical Data Forwarding in Vehicular Named Data Networks", *IEEE Communications Letters*, Vol. 21, No. 4, pp. 873-876, 2017. Available: 10.1109/lcomm.2016.2642194.

[10] O. Sahingoz, E. Buber, O. Demir, and B. Diri, “Machine learning based phishing detection from URLs”, *Expert Systems with Applications*, Vol. 117, No. 3, pp. 345-357, 2019. Available: 10.1016/j.eswa.2018.09.029.

[11] H. Le, Q. Pham, D. Sahoo, and S. Hoi, “URLnet: Learning a URL representation with deep learning for malicious URL detection”, *International Journal of High Performance Computing and Networking*, Vol. 1 No. 3, 2018. Available: arXiv preprint arXiv:1802.03162.

[12] H. Rakhshani and A. Rahati, “Snap-drift cuckoo search: A novel cuckoo search optimization algorithm”, *Applied Soft Computing*, Vol. 52, No. 03, pp. 771-794, 2017. Available: 10.1016/j.asoc.2016.09.048.

[13] G. Wang, A. Gandomi, X. Zhao, and H. Chu, “Hybridizing harmony search algorithm with cuckoo search for global numerical optimization”, *Soft Computing*, Vol. 20, No. 1, pp. 273-285, 2014. Available: 10.1007/s00500-014-1502-7.

[14] K. Jin, M. McCann, E. Froustey, and M. Unser, “Deep Convolutional Neural Network for Inverse Problems in Imaging”, *IEEE Transactions on Image Processing*, Vol. 26, No. 9, pp. 4509-4522, 2017. Available: 10.1109/tip.2017.2713099 [Accessed 30 June 2020].

[15] U. Acharya, S. Oh, Y. Hagiwara, J. Tan, and H. Adeli, “Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals”, *Computers in Biology and Medicine*, Vol. 100, No. 9, pp. 270-278, 2018. Available: 10.1016/j.compbiomed.2017.09.017.

[16] R. Alubady, S. Hassan, and A. Habbal, “Pending interest table control management in Named Data Network”, *Journal of Network and Computer Applications*, Vol. 111, No. 6, pp. 99-116, 2018. Available: 10.1016/j.jnca.2017.11.002.

[17] R. Shubbar and M. Ahmadi, “A Filter-Based Design of Pending Interest Table in Named Data Networking”, *Journal of Network and Systems Management*, Vol. 27, No. 4, pp. 998-1019, 2019. Available: 10.1007/s10922-019-09495-y.

[18] M. Majeed, S. Ahmed, and M. Dailey, “Enabling Push-Based Critical Data Forwarding in Vehicular Named Data Networks”, *IEEE Communications Letters*, Vol. 21, No. 4, pp. 873-876, 2017. Available: 10.1109/lcomm.2016.2642194.

[19] R. Alubady, S. Hassan, and A. Habbal, "Adaptive Interest Packet Lifetime Due to Pending Interest Table Overflow”, *Advanced Science Letters*, Vol. 23, No. 6, pp. 5573-5577, 2017. Available: 10.1166/asl.2017.7424.

[20] T. Nguyen, HL. Mai, R. Cogranne, G. Doyen, W. Mallouli, L. Nguyen, M. El Aoun, E.M. De Oca, and O. Festor “Reliable Detection of Interest Flooding Attack in Real Deployment of Named Data Networking”, *IEEE Transactions on Information Forensics and Security*, Vol. 14, No. 9, pp. 2470-2485, 2019. Available: 10.1109/tifs.2019.2899247.

[21] A. Benmoussa, A. el KarimTahari, C. A. Kerrache, N. Lagraa, A. Lakas, R. Hussain, and F. Ahmad, “MSIDN: Mitigation of Sophisticated Interest flooding-based DDoS attacks in Named Data Networking”, *Future Generation Computer Systems*, Vol. 107, No.6, pp. 293-306, 2020. Available: 10.1016/j.future.2020.01.043.

[22] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, “Named Data Networking: A survey”, *Computer Science Review*, Vol. 19, No. 2, pp. 15-55, 2016. Available: 10.1016/j.cosrev.2016.01.001.

[23] R. Hou, L. Fang, Y. Chang, L. Yang, and F. Wang, “Named data networking over WDN based optical networks”, *IEEE Network*, Vol. 31, No. 3, pp. 70-79, 2017. Available: 10.1109/mnet.2017.1600132.

[24] S. Bouk, S. Ahmed, D. Kim, and H. Song, “Named-Data-Networking-Based ITS for Smart Cities”, *IEEE Communications Magazine*, Vol. 55, No. 1, pp. 105-111, 2017. Available: 10.1109/mcom.2017.1600230cm.