**ISSN: 0067-2904**

# Detection and Mitigation of Cache Pollution Attack Using Popularity Variation in Information Centric Networking Based on SDN

**Marwa Kareem Sameer*, Mustafa Ismael Salman**
*Department of Computer Engineering, College of Engineering, University of Baghdad, Baghdad, Iraq*

Received: 13/2/2022 Accepted: 16/7/2022 Published: 30/3/2023

**Abstract**

Information centric networking (ICN) is the next generation of internet architecture with its ability to provide in-network caching that make users retrieve their data efficiently regardless of their location. In ICN, security is applied to data itself rather than communication channels or devices. In-network caches are vulnerable to many types of attacks, such as cache poisoning attacks, cache privacy attacks, and cache pollution attacks (CPA). An attacker floods non-popular content to the network and makes the caches evict popular ones. As a result, the cache hit ratio for legitimate users will suffer from a performance degradation and an increase in the content's retrieval latency. In this paper, a popularity variation mechanism in a CCN-SDN environment (PV-CSDN) is proposed to detect and mitigate CPA. PV-CSDN is based on observing the behavior of legitimate users to learn the normal traffic pattern and record the required threshold values. Two key parameters are used to achieve the threshold values: the popularity of the contents and the average rate of repeated requests for each interface in the router. The current traffic pattern will be compared with the predefined thresholds and if any variation in the traffic is occurred the attack is detected. The algorithm was able to detect the attack, and as a mitigation process, the controller will block the malicious interface to prevent any further degradation in the performance. The experiments show that PV-CSDN detects and prevents the attack effectively.

**Keywords:** Cache pollution attack, information centric networking, content popularity.

# الكشف عن هجوم تلوث ذاكرة التخزين المؤقت والتخفيف من حدته باستخدام تباين الشعبية في الشبكات المتمحورة حول المعلومات استنادًا إلىSDN

**مروة كريم سمير*, مصطفى إسماعيل سلمان**
قسم هندسة الحاسبات, كلية الهندسة, جامعة بغداد, بغداد, العراق

**الخلاصة**

الشبكات المركزية المعلوماتية هي الجيل التالي من هياكل الإنترنت مع قدرتها على توفير التخزين المؤقت داخل الشبكة الذي يجعل المستخدمين يستردون بياناتهم بكفاءة بغض النظر عن موقعهم. في ICN ، يتم تطبيق الأمان على البيانات نفسها بدلاً من قنوات أو أجهزة الاتصال. تكون ذاكرات التخزين المؤقت داخل

*Email: m.kareem1305@coeng.uobaghdad.edu.iq

الشبكة عرضة للعديد من أنواع الهجمات مثل هجوم تسمم ذاكرة التخزين المؤقت وهجوم خصوصية ذاكرة التخزين المؤقت وهجوم تلوث ذاكرة التخزين المؤقت .(CPA) يقوم المهاجم بإغراق المحتويات غير الشائعة إلى الشبكة ويجعل ذاكرات التخزين المؤقت تطرد المحتويات الشائعة نتيجة لذلك، ستعاني نسبة عدد مرات الدخول إلى ذاكرة التخزين المؤقت للمستخدمين الشرعيين من تدهور الأداء وزيادة وقت استجابة استرداد المحتوى. في هذه الورقة، تم اقتراح آلية تباين شعبية في بيئة CCN-SDN (PV-CSDN) لاكتشاف وتخفيف .CPA يعتمد PV-CSDN على مراقبة سلوك المستخدمين الشرعيين لمعرفة نمط المرور العادي وتسجيل قيم العتبة المطلوبة. يتم استعمال معلمتين رئيسيتين لتحقيق قيم العتبة: شعبية المحتويات ومتوسط معدل الطلبات المتكررة لكل واجهة في جهاز التوجيه. ستتم مقارنة نمط حركة المرور الحالي بالعتبات المحددة مسبقًا وإذا حدث أي تغيير في حركة المرور، فسيتم اكتشاف الهجوم. كانت الخوارزمية قادرة على اكتشاف الهجوم وكعملية تخفيف، ستحظر وحدة التحكم الواجهة الضارة لمنع أي تدهور إضافي في الأداء. تظهر التجارب أن PV-CSDN يكتشف ويمنع الهجوم بشكل فعال.

## 1. Introduction

Continuous progress of networking technologies and growth of network traffic puts huge pressure on current network architecture. Nowadays, users are interested not in where data comes from but which content they want. Consequently, network architecture shifts from host-centric to content-centric architecture by using a new paradigm [1], [2]. Information-centric networking (ICN) is a future internet architecture proposed to overcome some limitations of IP networks. The current internet paradigm uses end-to-end communication between nodes. For data to be transmitted over the network, the location of source and destination must be known. While ICN is a location-independent model, each content (e.g. a photo or a video) has a name, and users can provide access to these contents by using these names [3]. One of the most effective functions in ICN is in-network caching in which each node can store contents requested by consumers to serve any future requests. A caching mechanism includes two parts: a caching placement algorithm, which decides which contents to be stored in the router's cache, and a caching replacement algorithm, which decides which contents to be evicted from the cache when it is full [2]. ICN is an open environment, and its fundamental function is based on in-network caching. These attributes bring many security issues, such as cache pollution attacks (CPA), which can cause a large number of cache misses for regular users and cause performance degradation. There are two kinds of CPA: locality disruption attacks (LDA) and false locality attacks (FLA). In LDA, the attacker constantly requests non-popular content. In this way, the popular content will be pushed out of the cache. In FLA, the attacker keeps requesting the same set of non-popular contents to force the ICN caches to evict the popular and store the unpopular contents [4]. Today, networks are complicated and very hard to manage. Software-defined networking (SDN) makes the network programmable and almost auto-managed by decoupling the control plane from the data plane. Openflow, the most common implementation for the SDN data plane, suffers from many limitations. Although it works with many common protocols, it is protocol-aware and does not support new protocols [5]. From the data-plane side, Protocol Oblivious-Forwarding (POF) [5] and Programming Protocol-independent Packet Processors (P4) [6] is a common examples on the programmable data planes. POF implements a protocol-oblivious forwarding representative by a high-level language. It removes the protocol dependence by defining a generic instruction set. It is also possible to modify packet formation more than OpenFlow. The use of the POF programmable switch is flexible and it improves forwarding plane programmability by bringing in flow metadata, adding instructions, and so on [7]. In this paper, an ICN with a programmable data plane based on POF switching has been proposed to tackle the limitations of the OpenFlow switches in SDN. A detection and mitigation scheme has been proposed by deploying Content-Centric

Networking (CCN) over POF to implement a defense mechanism that will prevent the performance degradation of the whole network due to the CPA.
The CPA scenario consists of three phases: The investigation phase is in which a threshold value will be computed to define the CPA attack. A detection phase in which the threshold value will be compared with the current forwarding traffic. Then, in the prevention phase, an attack is detected, and the adversary interface will be defined as a malicious interface and blocked by the controller.
The main contributions of this paper are:
1. Apply a cache pollution attack on the CCN-SDN network and study the effects of such an attack on the cache with the use of the LRU cache replacement algorithm.
2. Design a detection algorithm, based on calculating the popularity of the forwarded traffic and the average repetition rate for each content in the POF programmable switch.
3. Apply a defense schema that uses the SDN controller to block the malicious interface and prevent any requests from the attacker.

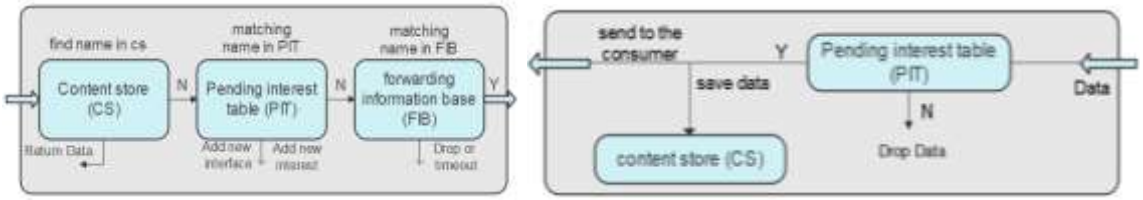
This paper is organized as follows: In section 2, the related work is discussed. Section 3 presents the system model and its main components. Section 4 outlines the PV-CSDN algorithm. The performance evaluation and the emulation results are discussed in section 5, followed by a conclusion in section 6.

## 2. System Model and General Design

This section describes the main model of the CCN-SDN architecture and its data structures that are used in content forwarding. However, a brief description of the main blocks of CCN and POF will be introduced first.

### *2.1 Content Centric Networking*

CCN is the most popular ICN architecture. It provides reliability and efficiency to content distribution by identifying contents in the network by their names rather than their sources and locations. It uses hierarchical names with a prefix and suffix like "ccnx:/networks/test1/version/segment1". It has two types of packets: an interest packet, which defines user interest carrying the name of the requested content; and a data packet, which contains the name of the requested content and the payload. CCN contains three main data structures: Content Store (CS), Pending Interest Table (PIT), and Forwarding Information Base (FIB). Once the interest packet arrives at a node, it checks if the data is stored in CS. If it exists, then the data will be forwarded downstream to the requester. If it does not exist in the cache, then the PIT table will be checked to verify if a matching name is found. If no pending name matches, the node will add a new entry to PIT. If there is an entry with the same name, then there are two possibilities: either the request came from the same interface, and in this case, the request will be dropped; or the request came from a different interface, and in that case, only the interface will be added and the request will be dropped. Next, the FIB table will be checked and a long prefix match will be applied to find a matching entry. If a match exists, then the interest will be forwarded to the next hop. Otherwise, the request will be discarded. The request will then be forwarded until it reaches a cache node or a producer to satisfy the request. Figure 1(a) shows the interest forwarding by CCN nodes. When the data packet returns from a cache node or a producer, it first verifies the PIT. If the name of the data does not exist, it means the data was not requested and it will be dropped. If a pending request exists, then the data will be cached in the node and an entry will be added to the CS. Finally, the data packet will be forwarded downstream to all interfaces in the PIT following the same path the request came from. Below is the forwarding process of the data packet as illustrated in Figure 1(b).

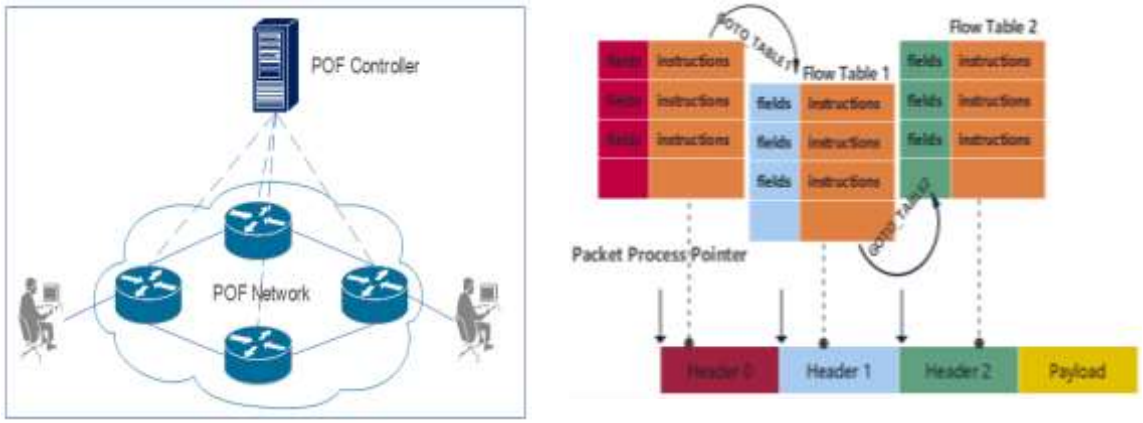



**(a)** **(b)**

**Figure 1:** The forwarding process of CCN node. (a) The forwarding process of interest packet. (b) The forwarding process of data packet.

*2.2 Protocol-Oblivious Forwarding*

POF is the enhancement of the SDN OpenFlow forwarding data plane. The objective of this enhancement is to improve the programmability of the SDN. It enables forwarding services and supports new protocols without any modification to the network elements. The switch only needs to extract the search keys, which are the {offset, length} tuples, lookup the flow tables, and if a match occurs, the corresponding instructions will be implemented. The offset of the search keys represents the number of skipped bits in the packet from the current position, and length is the number of bits that a key should have started from the offset position [5].




**(a)** **(b)**

**Figure 2**: Overview of POF (a) a general view of SDN network based on POF. (b) The packet forwarding process in POF.

Figure 2(a) shows a standard SDN network that consists of a POF controller to control and manage the network and POF switches in the data plane. Figure 2(b) shows the procedure of packet forwarding, where the packet has three layers of protocol headers, from header 0 to header 2. Each header is handled by a flow table, which contains a set of matching fields and a set of instructions. If a match occurs, the corresponding instructions will be executed. When the matching for the specified header is completed, a GOTO instruction is used, and the packet process pointer is moved to point on the next header, and the process is repeated [8].

*2.3 CCN-SDN architecture*

In this paper, the implementation of CCN-SDN of [9] is used. Figure 3 illustrates a general view of the CCN-SDN architecture. It is composed of three main components:

- The POF controller: written in Java and communicates with multiple POF switches, and it is responsible for managing and controlling the network, identifying the ICN protocol,

setting the forwarding rules, and adding or deleting the cache function in the switches. Figure 4 shows the functions of the controller.

- POF switches: written in C language, they are the data plane part of the network, and their main function is forwarding the traffic.
- ICN nodes: consist of consumers, which issue interest packets, and providers, who provide data packets to the consumers. It connects to the POF switches.

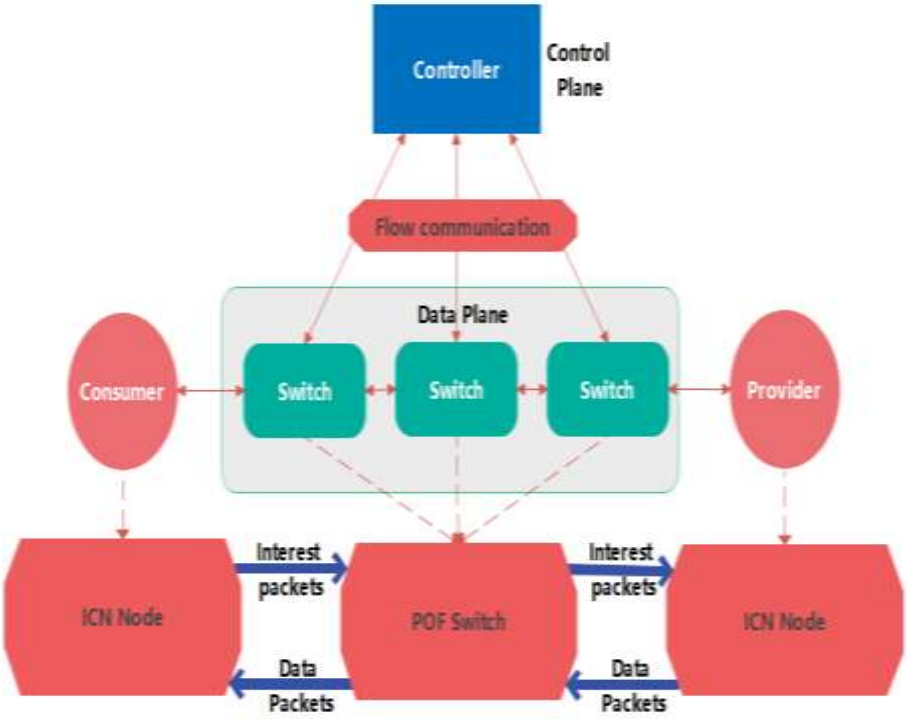



**Figure 3:** System Architecture.

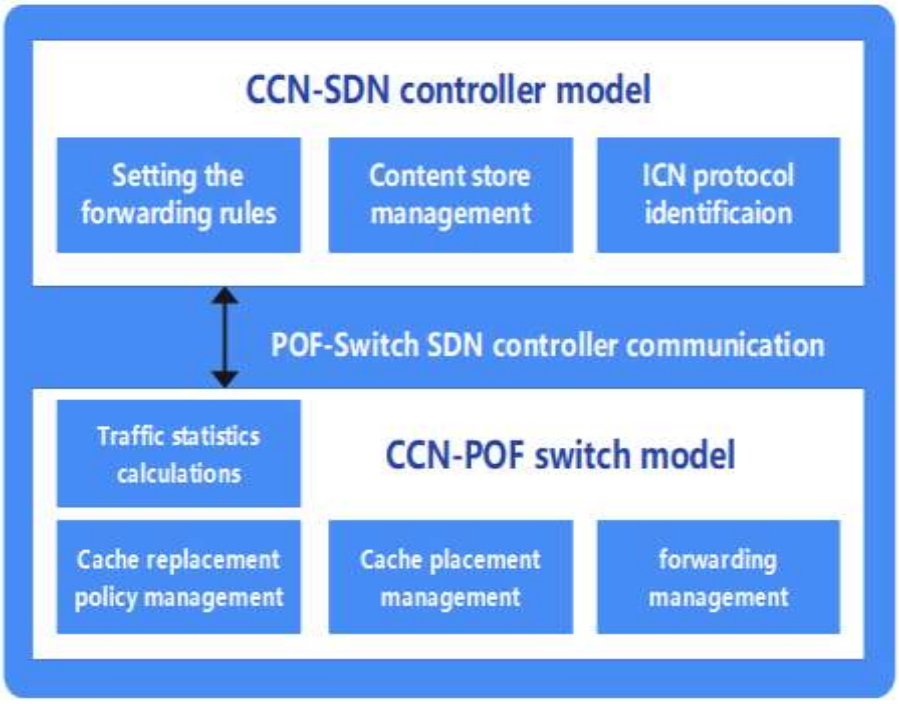



**Figure** 4: CCN-SDN System functions

The POF switch is also responsible for caching the contents in its content store, evicting content from the cache when the cache is filled, and calculating traffic statistics such as the number of requests received for each content, cache hit ratio, the popularity of each content, which will be used in detecting CPA, and so on. Figure 4 shows the functions of the controller and the switch in the CCN-SDN system.

The main data structures in the switch are shown in Figure 5.

1. The CS table consists of a cache placement algorithm that enables the switch to save a copy of data in it. The default caching algorithm is to leave a copy everywhere (LCE), which saves any content that comes into the cache. When the cache is out of free space, the content will be deleted using a replacement algorithm so that new content can be stored in the cache. The default cache replacement algorithm is the Least Recently Used (LRU), which removes the content that has the longest time period of not being requested.
2. The FIB table contains the forwarding rules that are installed in the switch after it connects to the controller. It helps in forwarding the requests to providers.
3. The PIT table helps the data packets return to consumers by recording the name of the request and the port from which they came.
4. The DDoS table contains the traffic statistics such as the popularity value of contents, the time of receiving each content, the total number of requests the switch received, and the number of cache hits.

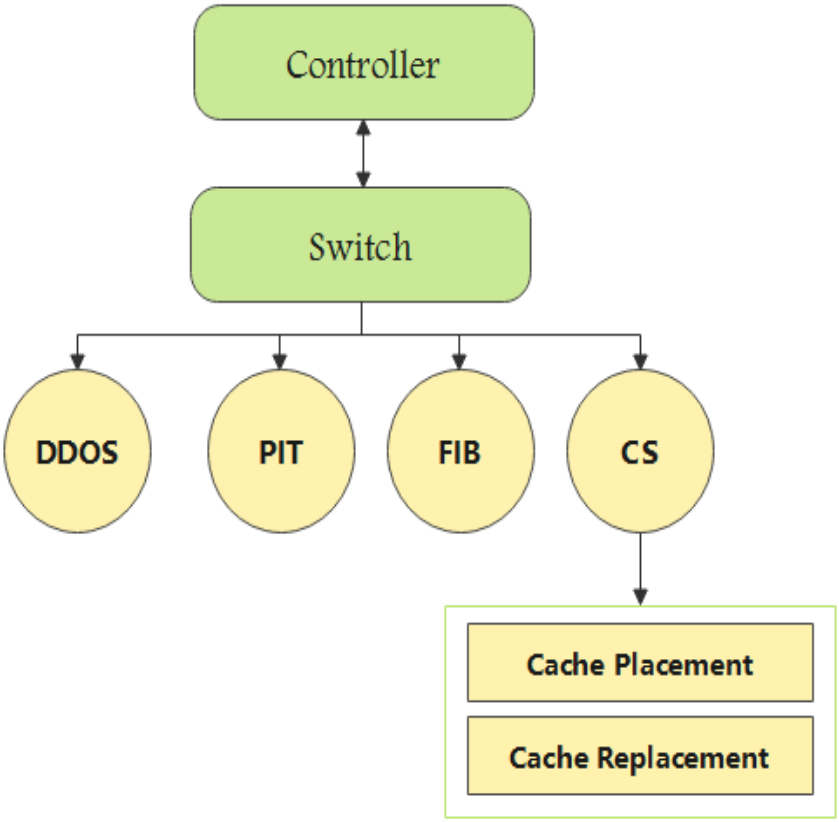



**Figure** 5: POF-switch structure in the CCN-SDN system

## 3. Related Work

There is a large volume of published studies describing the CPA for the internet such as the studies in [10], [11], however, limited works have been proposed for the detection and mitigation of such attacks in ICN.

In a large longitudinal study, Karami et al. [12] proposed a cache replacement method called ANFIS. It is an integration of fuzzy inference system and neural network system. The input is the inherent characteristics and statistical data of the cached contents, and the result is a value that represents the type of the content, i.e., FLA, LDA, or healthy. It extracts six features from the input data: the standard deviation of the content's access frequency, the time in which each content remains in the cache, the last retrieval of the content, the variance of the entire population, the popularity of the contents; and the hit ratio of each content in the cache. The algorithm is complex and puts too much computation overhead on the routers.

In contrast to [12], Zhang et al. [13] suggested a non-collaborative cache decision-making scheme, called IFDD that combines the locality and popularity of contents to enhance the router cache and mitigate CPA. Popularity is calculated by counting the number of requests for each content, while the locality is obtained by finding the number of interest packets for each content at each interface in the router and then computing the coefficient of variation (CV) for all the requested contents. The CV and popularity are used to find a decision matrix to decide which contents should be cached. This method holds too many calculations to apply the caching decision matrix, and these calculations are increased when the number of contents increases.

In the same vein, Yao et al. [14] proposed a mitigation and prevention method based on the gray prediction model to predict the popularity of the cached contents by obtaining the popularity of each content in the cache. This popularity should be accompanied by other three factors, which are the request ratio, the standard deviation of the request interval of the contents, and the variance of the repeated interests. A CPA is detected if the predicted value of the content differs from the computed one. The defense mechanism is handled by decreasing the popularity of the suspicious content to mitigate the CPA damage.

A seminal study in this area is the work of Chen et al. [15], who proposed a mechanism called CCP, which uses the content popularity statistics as a historical parameter and predicts the future popularity of the content using the multiple linear regression fitting scheme. As a mitigation process, a probabilistic caching decision strategy has been used; also, it is adjusted according to the abnormality of the contents that arrive at the cache. This work is applied only to a small scale mobile network. In addition, it causes an increase in the computation overhead. Other than the previously mentioned works, Rani et al. [16] proposed a secure framework based on Fuzzy Restricted Boltzmann Machine to detect and defend against the abnormal behaviors of CPA in ICN networks. Besides, a replacement policy based on a reward value is used to suppress the suspicious contents by giving a lower reward value so that they will not be cached. The research did not indicate the effect of the attack on the system. The author did not mention how the data set was collected.

In a study conducted by Yao et al. [17], a clustering scheme to detect the CPA is proposed. By grouping the content requests into popular and non-popular clusters, the algorithm was able to distinguish the suspicious from the normal requests. The research also explains the state when a number of requesters become interested in some unpopular data. In this scheme, the defense mechanism is based on identifying the malicious requests, adding them to an attack table, and sending a broadcast warning to neighbors to not cache them. This process depends on the identification of the requests from the attacker. That means, the malicious contents will be cached in case of misidentification. This will degrade the cache hit ratio of the router and decrease the network performance.

Kamimoto et al. [18] proposed a method to protect the cache against CPA. The protection schema is based on the hierarchal prefixes of the content name. Each router will identify the set of prefixes that are requested by the adversary in order to protect its cache. After identifying the malicious prefix, the cache will delete all the suspicious contents and avoid caching any contents in the blacklist. If the attacker uses the popular prefixes to carry out the attack, then the name prefixes that are included in the blacklist will be useless.

In addition, Xie et al. [19] presented a method called cache-shield to improve the robustness of the cache against the LDA. This method evaluates a shield function that prevents the caching of non- popular content. The interest packet will be forwarded and its data packet will be cached only if the shielding function returns true; otherwise only the name of the content will be saved. This method has not shown whether it can be applied to large-scale networks and it has no effect on the FLA.

A lightweight detection method is presented by Conti et al. [20]. This approach consists of two phases: the learning phase and the detection phase. In the first phase, a traffic sample is taken to study the patterns of normal users and record a threshold value. In the second phase, the threshold value is compared with a traffic sample of the current distribution. This method did not take the whole content domain to study the traffic pattern, and it lacked a prevention mechanism to deny the malicious requests and stop the performance degradation.

## 4. PV-CSDN Algorithm

This section proposes the general problem and lists some assumptions needed for applying the CPA to the network. It also describes the procedure of applying the attack and how to use the PV-CSDN algorithm to detect and mitigate it.

### *4.1 problem statement*

CPA happens in the above network environment when an illegitimate user damages the stability of the cache by sending a bunch of unpopular content. The availability of caching will be reduced as well as increase the latency of legitimate users for acquiring the data. Therefore, a detection approach is required in order to prevent further damage to the network. To apply a CPA, the attacker monitors the network and records the requested contents. He/she will inject the network with a small set of non-popular content to increase the cache misses for legitimate users. Applying an effective attack requires using two parameters. One is the attack power ratio ($\zeta$) which is the ratio between the number of interest packets issued by the attacker and the number of interest packets issued by legitimate users, and the other is the attack range ($\varphi$) which represents the number of content objects attacked. The FLA can be constructed easily based on these two parameters.

Before defining the proposed countermeasures, some assumptions, which are similar to [14], [17], and [20], are defined:

1. The attacker can issue interest packets by compromising one or a set of consumers.
2. The attacker does not have any special privileges, such as changing the configuration of any caching policy.
3. The attacker does not have the ability to generate any new content on the network.
4. The attacker can request any content from legitimate providers.

### *4.2* Proposed Countermeasures

This section presents the popularity variation in content-centric based on the software-defined networking (PV-CSDN) algorithm, which is the proposed solution to preserve the

caching system by detecting and mitigating the CPA and maintaining efficient content delivery in the network. The most common notations used are listed in Table 1 along with their corresponding terminologies.

**Table 1:** Frequently Used Notions.

| Symbol | Description |
|---|---|
| C | The set of contents. |
| Ci | The ci$^{th}$ content in C. |
| P(ci) | The popularity of the ci$^{th}$ content |
| SV | The amount of variation. |
| μ | The threshold value. |
| ζ | The attack power ratio. |
| φ | The attack range. |

The popularity distribution of the requested content can be obtained by calculating the following parameter:

- The popularity P(ci) associated with each content ci, is obtained as:

$$P\_new(ci) = \alpha^{\Delta t} + P\_old(ci) \quad (1)$$

- Where P_new(ci): the updated popularity, Δt: the time interval between two requests for content ci in C, α**:** is the popularity constant**,** P_old(ci): the old popularity for ci.
- The Ps_new (ci) is the calculated popularity during a sample number of requests.

$$Ps\_new(ci) = \alpha^{\Delta t} + Ps\_old(ci) \quad (2)$$

Where Ps_new(ci): the popularity for ci during a sample number of requests,

- The SV is the amount of difference between the sample popularity of content **ci** and the total popularity P_new(ci) for the same content ci.

$$SV = \sum_{ci \in C} \left(\frac{Ps\_new(ci) + Cs(ci)}{S} - \frac{P\_new(ci) + C(ci)}{T}\right) \quad (3)$$

Where SV: the amount of variation for the current sample, S: the sample size, T: the total number of requests, Cs (ci): the number of requests for ci during the sample period, C(ci): the total number of requests for ci.

The approach takes a number of samples (SVs), finds their mean plus α times their standard deviation to obtain a threshold value represents the maximum amount of variation a traffic can reach.

$$\mu = \frac{\sum_{i=0}^{N} SVi}{N} + \alpha \sqrt{\frac{1}{N}\sum_{i=0}^{N}\left(SVi - \frac{\sum_{i=0}^{N} SVi}{N}\right)^2} \quad (4)$$

Where N: the number of samples, μ: the threshold value.

The technique of calculating the threshold value is similar to [20], where the threshold has been calculated by using only a set of contents, not the whole content domain.

The threshold value that represents the popularity distribution may not be enough to provide an accurate detection to CPA since the legitimate users may suddenly become interested in some non-popular content and that will cause a false positive state. To avoid this, another traffic characteristic has been used. The average rate of repeated requests for all interfaces in the router is computed by finding the variance of requests for each interface.

$$\text{Mean[pi]} = \frac{\sum_{ci \in C} \text{count (ci)}}{C} \quad (5)$$

Where Mean [pi]: the mean value for all requests came from interface pi, C: the contents of one interface.

$$\text{V [pi]} = \sum_{c\, i \in C}((\text{count (ci)} - \text{Mean [pi]})^2 \,/\, \text{C}) \quad (6)$$

Where V [pi]: the variance of all requests came from interface pi.

The requests of the attacker in CPA will follow the Zipf-like distribution [21] as the legitimate users. In LDA, the adversary will request all contents with an equal probability, while in FLA, the adversary will continuously issue requests for only a subset of non-popular contents [17].

*4.3 CPA Scenario*

A general description of how the algorithm works is as follows (shown in Figure 6):

1- When a router receives an interest, it updates the total number of requests, the number of interests for the same content, the total popularity, the sample popularity of each content, and the variance of the corresponding interface.

2- When the number of requests reaches the sample size, it computes the amount of popularity variation for the current sample and updates the threshold value for the content distribution and the variance of each port in the router.

3- Based on the results of the popularity variation value and the variance, it can be determined whether the network is under attack or not.

4- If an attack is detected, a defense process will be launched and the malicious interface will be blocked by the controller.

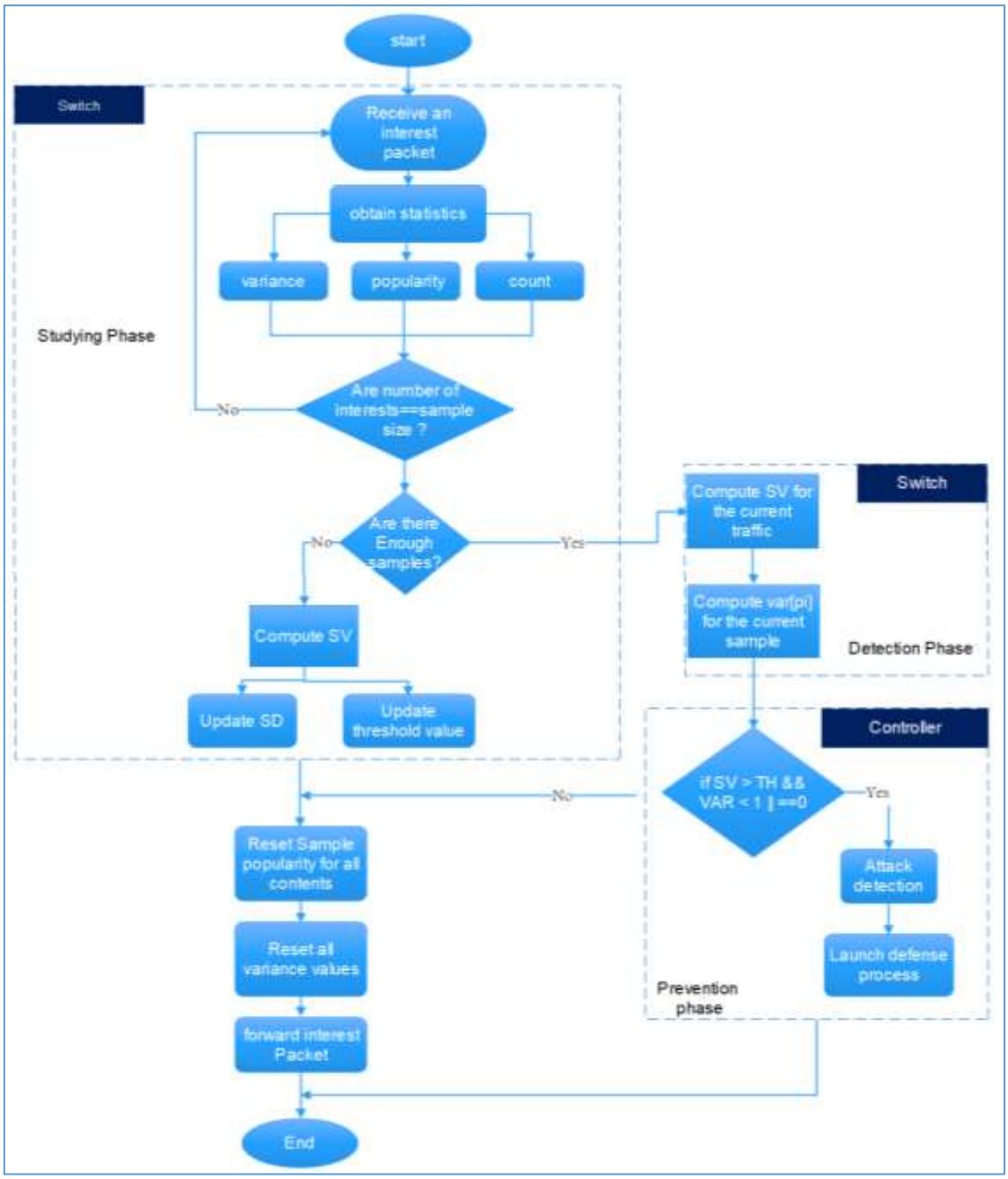



**Figure 6**: Flow Chart of CPA Scenario

*4.4 Detection of CPA*

The CPA scenario consists of three main phases. The study, the detection, and the prevention phase. To detect CPA, the following scenario is followed:

In the studying phase, the normal behavior of legitimate users will be studied to compute the required values of threshold and interface variance in algorithm 1. When an interest packet is received, some parameters will be updated in the switch. If the name of interest is in the DDoS_table, then the corresponding popularity values will be updated; otherwise, a new entry will be added and the popularity will be calculated; then the variance of the corresponding interface will be updated on lines 1 to 11.

Line 12 will check whether the total_req has enough requests to invoke the studying phase.

If the studying function is invoked, the threshold value will be updated and sent to the controller. Then all the variance values of all interfaces in router R, as well as the popularity sample, PS_new and sample_ count (Cs), for all contents, will be reset to start computing their values from zero on lines from 13 to 18. When enough samples are obtained, the studying phase is finished, and the detection phase will be launched to compute the sample variation of the current traffic on line 19.

If the interest packet is satisfied from a cache node, then a copy will be returned to the consumer. Otherwise, it will be forwarded to the provider to get the best available content from 22 to 25. When the content is returned to the consumer, a copy will be stored in the cache of all nodes in the path. If the cache is full, the content will be deleted according to the LRU cache policy. If the cache is not full, it will save a copy of the content on 26 to 29.

| *Algorithm 1* | |
|---|---|
| *Input:* | S = 10000 request. |
| *1.* | Incoming interest packet at router R from interface $p_i$ |
| *2.* | Update total number of requests in router R [total_req = total_req +1] |
| *3.* | **If** name of interest packet in ddos_table |
| *4.* | Update [ t_old(ci), t_new(ci), ts_old(ci), ts_new(ci)] |
| *5.* | Update [ P_old(ci), P_new(ci), Ps_old(ci), Ps_new(ci), C(ci), Cs(ci)] |
| *6.* | **Else** name of interest packet not in ddos_table |
| *7.* | Add new entry to ddos_table |
| *8.* | Record [ t_old(ci), t_new(ci), ts_old(ci), ts_new(ci)] |
| *9.* | Record [ P_old(ci), P_new(ci), Ps_old(ci), Ps_new(ci), C(ci), Cs(ci)] |
| *10.* | **End if** |
| *11.* | Update variance value of ci in (updating variance value)<br>Updating_variance (total_count[pi], count[ci]) |
| *12.* | **If** (total_req mod S) == 0 then |
| *13.* | **If** Studying_phase(total_req, S, P_new[C], PS_new[C]) then<br>// Studying phase is not completed yet |
| *14.* | Send the threshold value to the controller. |
| *15.* | **For** all ci in C |
| *16.* | PS_new(ci )= 0 , PS_old(ci ) = 0, Cs(ci) = 0 |
| *17.* | Reset all variance values. |
| *18.* | **End for** |
| *19.* | **Else:**<br>// studying phase is completed<br>detection_phase(total_count, S) |
| *20.* | **End if** |
| *21.* | **End if** |
| *22.* | **If** interest packet satisfied from cache then return data to consumer |
| *23.* | **Else** forward the interest packet |
| *24.* | **End if** |
| *25.* | Get the best available content |
| *26.* | **If** cache is full |
| *27.* | Evict content from cache according to LRU cache policy and save the new content |
| *28.* | **Else** save a copy of content in cache and return it to consumer |
| *29.* | **End if** |

Algorithm 2 describes the variance update when a new interest packet comes from interface pi. The total_count for interface pi and the count of (ci) will be updated. By updating, the mean value for pi and the total variance value for all the content population in lines 1 to 6.

| **ALGORITHM 2** | **UPDATING VARIANCE** |
|---|---|
| **1.** | Update total number of requests [total_count[pi] = total_count[pi]+1] |
| **2.** | Update number of requests for content ci [count(ci) = count(ci)+1] |
| **3.** | mean[$p_i$] = total_count[$p_i$] / C[pi] |
| **4.** | **For** all $c_i$ in C from $p_i$ |
| **5.** | Variance[pi] = $\sum_{i=1}^{C}(\text{count(ci)} - \text{mean[pi]})\mathbf{2}/\text{C}$ |
| **6.** | **End for** |

In algorithm 3, the studying phase consists of periodically taking a sample (SV) to calculate a threshold value. The sample size (S) has been selected to be 10,000 content objects, which is the number of items in the content domain. The number of samples has been selected to be 15, since the threshold value will become stable after 15 traffic samples. The studying phase will be stopped then and new invocations will return false on lines 1 to 9. When the learning phase is complete, the threshold and the variance values will be sent to the controller.

| ***Algorithm 3*** | ***The Studying Phase (Threshold Calculation)*** |
|---|---|
| ***1.*** | **If** total_req/S < 15 |
| ***2.*** | **Return false** |
| ***3.*** | **For** all ci in C |
| ***4.*** | Sample_popularity(ci) = PS_new(ci)+ Cs(ci)/S |
| ***5.*** | Total_popularity(ci) = P_new(ci) )+ C(ci)/total_req |
| ***6.*** | SV = SV + \|(Sample_popularity (ci) - Total_popularity (ci) )\| |
| ***7.*** | **End for** |
| ***8.*** | Update μ according to eq. 4 |
| ***9.*** | **Return true** |

In algorithm 4, the detection phase will be started after the studying phase finishes. It will compute the sample popularity variation of the current traffic on lines 1 to 5. From 6 to 10, the rate of repeated requests for all ports in the router will also be computed. Resetting PS_new and the variance values for all contents on lines 11 to 14, then the result and the repeated rate for all interfaces will be sent to the controller to check for the attack on line 15.

| Algorithm 4 | The Detection Phase |
|---|---|
| **1.** | **For** all ci in C |
| **2.** | Sample_popularity(ci) = PS_new(ci) +Cs(ci) / S |
| **3.** | Total_popularity(ci) = P_new(ci) +C(ci) /total_req |
| **4.** | SV = SV + \|(Sample_ popularity (ci) - Total_ popularity (ci) )\| |
| **5.** | **End for** |
| **6.** | **For** all pi in router R |
| **7.** | **For** all ci in C from pi |
| **8.** | Variance[pi] = $\sum_{i=1}^{C} variance(ci)/C$ |
| **9.** | **End for** |
| **10.** | **End for** |
| **11.** | **For** all ci in C |

| | |
|---|---|
| 12. | PS_new(ci )= 0 , PS_old(ci )= 0, Cs(ci)= 0 |
| 13. | Reset the variance values of all ports in the switch. |
| 14. | **End for** |
| 15. | Send statistics to controller |

*4.5 Defense against CPA*

Algorithm 5 illustrates the prevention phase; the controller will check the statistics values that are sent from the switch. When SV is more than the predefined threshold value and the variance of any interface is equal to zero or less than one, then there is an attack on the network. The defense process is based on identifying the interface where the malicious content came from. After that, the controller will block it to prevent any further damage to the network.

| *ALGORITHM 5* | THE PREVENTION PHASE |
|---|---|
| *1.* | **For** each interfaces in router R |
| *2.* | **If** (SV > μ) && ( Var [$p_i$] == 0) |
| *3.* | Result = true - LDA → Block interface pi |
| *4.* | **Else if** (SV > μ) && (1 >Var[$p_i$] > 0) |
| *5.* | Result = true – FLA → Block interface pi |
| *6.* | Else |
| *7.* | Result = No attack. |
| *8.* | **End if** |

## 5. Performance Evaluation

The experiment has been evaluated in Mininet [22], an open-source emulation environment, which has the ability to implement scenarios similar to a real experiment. The implementation of the SDN-CCN architecture is based on the POF controller and POF switches. Each switch is linked to the controller logically. CCNX [23] is the official implementation of the CCN architecture and is running on the ICN nodes as consumers and providers. Two virtual machines are used to implement all the scenarios in Virtual-Box. The first three VMs embedded on Debian OS are Mininet, POF-switch, and CCNX. The second VM runs the POF controller in Eclipse on Ubuntu 14.06 LTS. All the parameters used are listed in Table 2.

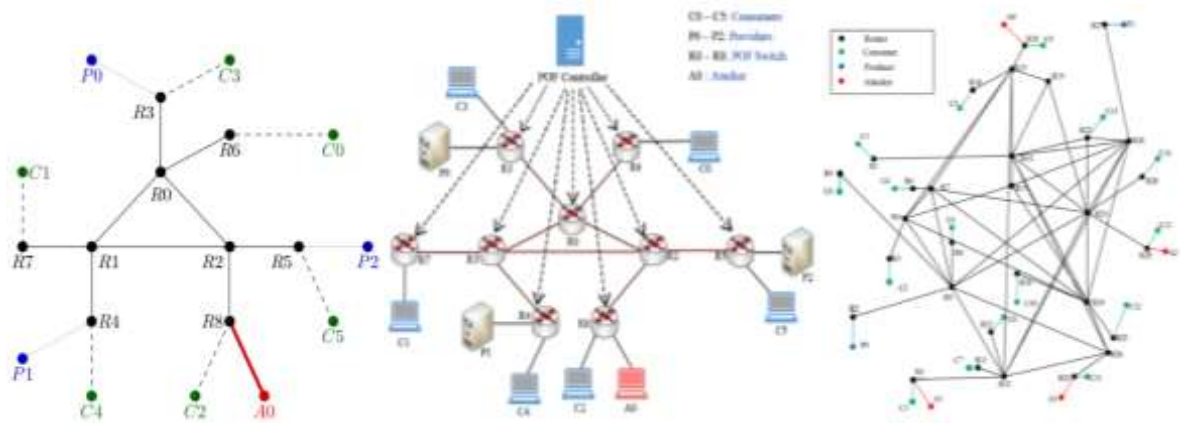
**Table 2:** list of the simulation parameters

| *SIMULATION PARAMETERS* | *VALUE* |
|---|---|
| Number of contents | 10000 |
| CS Size | 100 |
| PIT Size | 1200 |
| Legitimate Request Rate | 50 interests/second |

The domain of content objects is static; it contains 10,000 content items. The router CS is 1% of the number of contents. On the other hand, the cache placement policy is LCE while the default cache replacement policy is LRU, which is a commonly used in-network cache. The topology is the XC topology, as illustrated in Figure 7. In XC, there is 1 attacker, 6

legitimate users (CCN consumers), 3 providers, a POF Controller, and 9 POF switches. The second topology is DFN, which will be used only during the comparison with other works.

The complete simulation scenario consumes 4 hours. During the first two hours, all interest packets are issued by legitimate users and follow a Zipf-like distribution. Then, the adversary will start the attack at the second hour.




(a) (b) (c)

**Figure 7:** Network topologies (a) the XC topology. (b) XC topology with CCN-SDN. (c) DFN topology.

*5.1 Performance Metrics*

The comparison is made between the PV-CSDN scheme, ANFIS [12], and DDCPC [17]. The PV-CSDN scheme detects the CPA by monitoring the popularity variation of the forwarded traffic and the rate of repeated requests from each interface. ANFIS uses the fuzzy inference system with a neural network and compares with a goodness value so that those with a lower value will be considered as fake. DDCPC detects the CPA by clustering the interest distribution. In the comparison, the following metrics have been used:

1- Cache Hit Ratio: The ratio between the number of cache hits and the total number of requests.

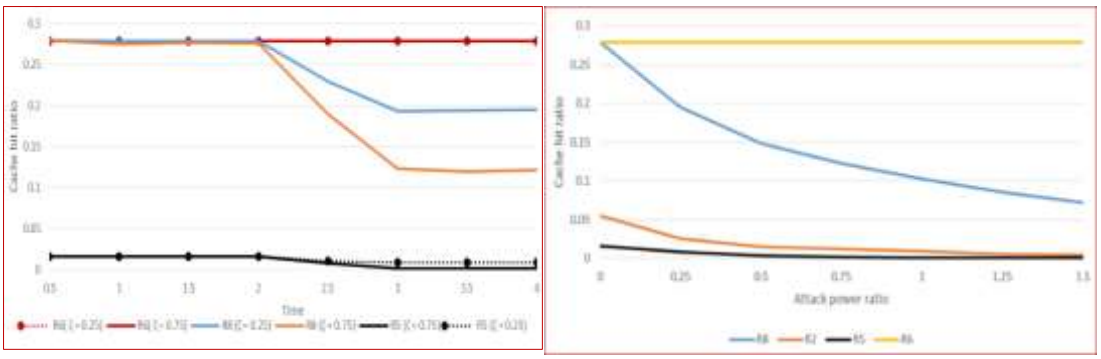
2- Average Hop Count: The average number of hops a user gets to receive the data packet.

*5.2 Impact of CPA*

In this section, the effect of CPA damage has been evaluated under the least recently used (LRU) cache replacement algorithm, which evicts the data in the cache that has the longest time of not being accessed. To demonstrate the effect of LDA and FLA in the network, three types of routers are chosen: the router connecting to a normal consumer (R6), the router connecting to an attacker (R8), and the router connecting in the upstream path (R5), to evaluate the effect on cache hit ratio under the attack. Two consumers, C0 and C2, were chosen to evaluate the effect on the average hop count under the attack. One attacker (A0) is considered as illustrated in Figure 7. The attack power ratio ($\zeta$) ranges from 0.25 to 1.5 in a step of 0.25, and the attack range ($\varphi$) will be from 50 content to 400 content.

*5.2.1 Impact under LDA*

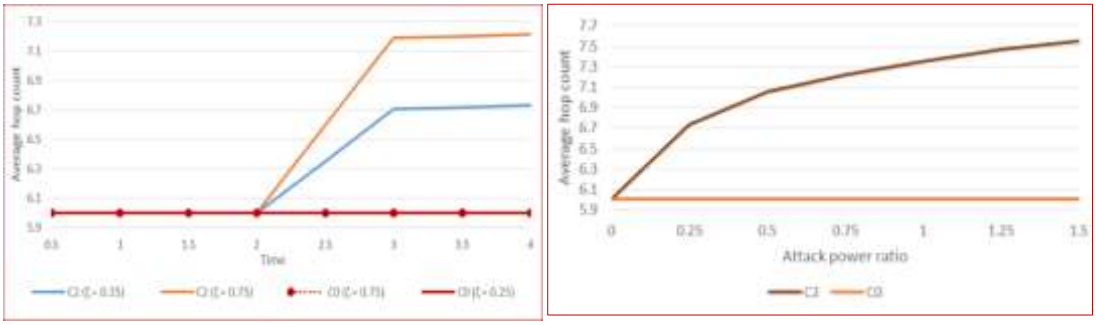
Figure 8 shows the impact of LDA on cache hit ratio. In Figure 8 (a), before the attack starts, the hit ratio of R8 is stable at 0.279. After launching the attack in the 2nd hour, the hit ratio of R8 decreases. This is because of its direct connection to the attacker A0 so that all the requests from A0 will be satisfied by the provider, which leads to a cache miss in R8. The hit ratio of R5 also decreased, while in router R6, the hit ratio was not affected because it is a far distance from A0. As shown in Figure 8 (b), the hit ratio of R8 continuously decreases as the power ratio increases because the number of cache misses increases when the requests from A0 increase, and all of them will be satisfied by the provider. As a result, the hit ratio in the switch degrades. It became 0.0.0721 with $\zeta = 1.50$, a 74.1% decrease. R2, and R5 also continue to decrease until they become zero.



(a) (b)

**Figure 8:** Hit ratio under LDA in LRU (a) Cache hit ratio with respect to time (b) Cache hit ratio with respect to attacking power ($\zeta$)

Figure 9 shows the impact of LDA on the average hop count of C0 and C2. In Figure 9 (a), the average hop count of consumer C0 who is not connected to any attacker remains stable. As for consumer C2, who is connecting to A0, the hop count will be increased once the attack is launched because all the requests from A0 will be satisfied by the provider and that will increase the total number of hops. Figure 9 (b) shows that the hop count will increase continuously as long as the attacking power increases, which leads to an increase in the number of requests that take their response from the provider. The hop count can be up to 7.54 with $\zeta = 1.50$, an increase of 25.66%.



(a) (b)

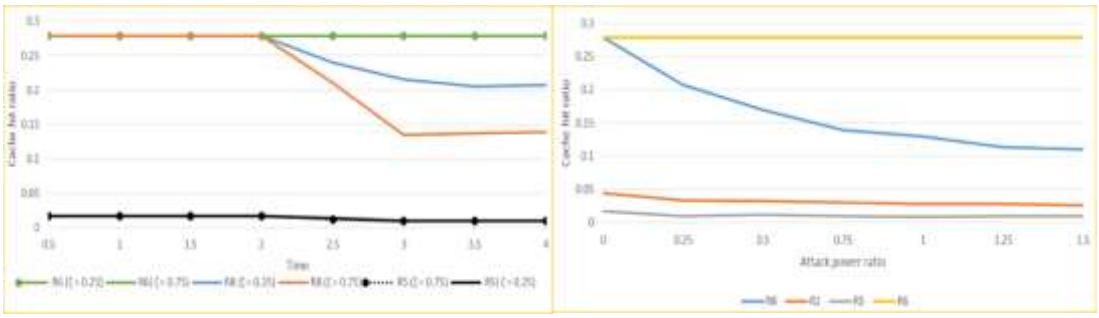
**Figure 9:** Average hop count under LDA in LRU. (a) Average hop count with respect to time. (b) Average hop count with respect to attacking power ($\zeta$).

*5.2.2 Impact under FLA*

In FLA, it is assumed that the attacker only requests content that is not widely used on the network. The attack consists of requesting a subset of the least-requested contents that are placed on the long tail of the Zipf-like distribution. The attack range (φ) is set to be 300 non-popular items.

Figure 10 shows the impact of FLA on the hit ratio. Specifically, Figure 10 (a) shows that the hit ratio of R8 decreases as soon as the attack starts at the 2$^{nd}$ hour, while the hit ratio of R6 stays constant at 0.279 because it is not on the attack path and it is far from the attacker A0.

The hit ratio of R5 degrades slightly and becomes almost zero. Figure 10 (b) shows the hit ratio with respect to the attack power ratio. The hit ratio of R8 decreases from 0.279 at no attack to 0.109 when ζ is 1.50 and a decrease of 60.9%. This is due to its direct connection to the attacker A0, which makes some requests that take its data from R2, R5 or from the provider. This will cause a cache miss in R8 and decrease the hit ratio. The hit ratio R2 degrades a little when the attack starts and remains relatively stable when the power ratio increases. The same is true for R5, which decreases a little bit and becomes stable when the power ratio increases. That's because some requests from an attacker will result in a cache hit in R5 and R2.

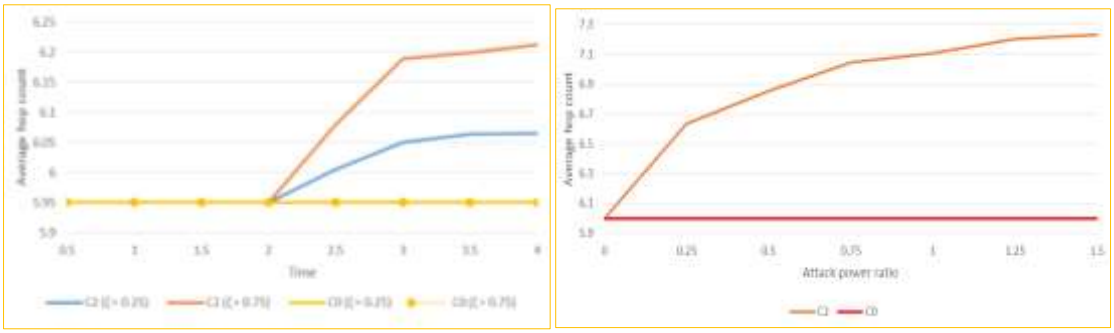


(a) (b)

**Figure 10:** Hit ratio under FLA in LRU. (a) Cache hit ratio with respect to time. (b) Cache hit ratio with respect to attacking power (ζ).

Figure 11 shows the average hop count due to FLA. For the normal consumer C0, the hop count remains relatively stable because it is not connected to any attacker. The hop count of C2 begins to increase once the attack is launched. As shown in Figure 11(a), it increases by 17.3% during the second hour when the attack power is 0.75.

11(a). While in Figure 11 (b), the hop count becomes higher as ζ increases. In other words, when ζ is 1.50, the average hop count becomes 7.23, with an increase of 20.5%. This behavior is due to the increase in the number of requests that are not taking their data from R8. Instead, they are taking it from R2 or R5 or the provider, which is three hops away from the consumer.




(a) (b)

**Figure 11:** Average hop count under FLA in LRU. (a) Average hop count with respect to time. (b) Average hop count with respect to attacking power (ζ).

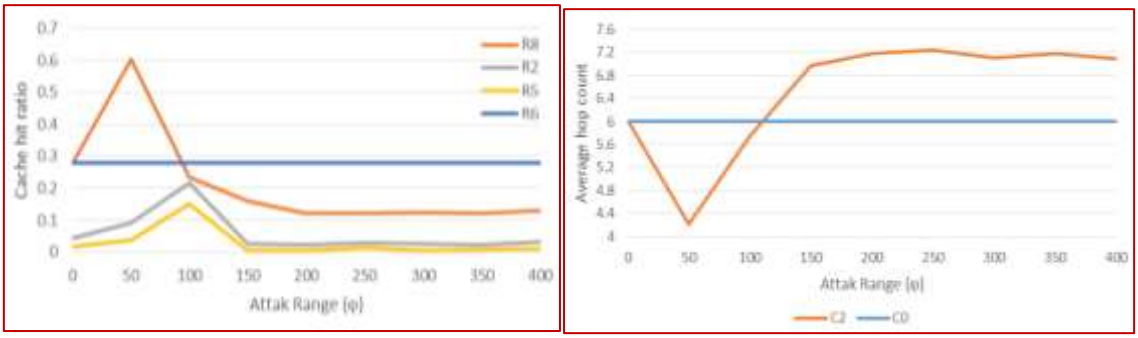
*5.2.3 Impact on number of contents attacked (range) under FLA*

Figure 12 shows the impact of different values of attack range φ with an attack power ratio is set to 1. Figure 12 (a), shows the effect on the cache hit ratio of different routers. When the range is less than the first router's cache size, the attack will not propagate and all malicious interests will be satisfied from the first router's cache, which is why the hit ratio of R8 has increased when it is 50.

When φ is100, the hit ratio of R8 starts to decrease while the hit ratio of R2 and R5 starts to increase because the attack propagates to R2 and R5 and their cache hits are increased while the number of hits in R8 starts to decrease.

When the attack range increases, the hit ratio starts to decrease until it gradually becomes stable for all routers on the attack path because the attack will propagate and some malicious requests will get their data from the provider. The hit ratio of R6 is stable because it is four hops away from the attacker.

In Figure 12 (b), the hop count of consumer C0 is stable on R6 because it is not connected to the same router as A0 and that is a far distance from the attack path. The hop count of consumer C2 drops to 4.2 when φ reaches 50 because all of the attacker's requests are satisfied by the first router, which is only one hop away from C2. However, the hop count begins to increase when φ increases, and finally, it stabilizes when φ is 200 or greater.




*(a)* *(b)*

**Figure 12:** Impact of attack range in FLA. (a) Average hop count with respect to subset range. (b) Average hop count with respect to subset range.

*5.4 Performance comparison*

In this section, the PV-CSDN algorithm has been compared with DDCP and ANFIS in terms of cache hit ratio and average hop count. Since the results of DDCP and ANFIS are applied in the DFN topology as shown in Figure 7 (c), the comparison with the PV-CSDN algorithm will be applied in the DFN as well. R5 is used to show the cache hit ratio under attack because of its direct connection to the attacker A3 and C3 to evaluate the average hop count, as shown in Fig 13.

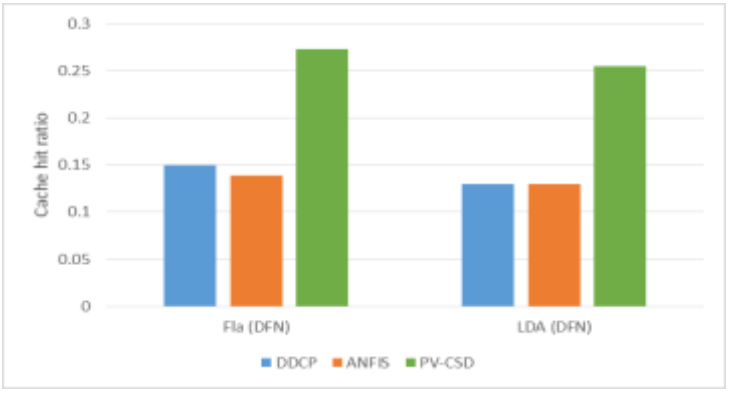



**Figure 13:** Hit ratio under CPA

PV-CSDN outperforms DDCP and ANFIS in FLA by 82% and 96%, respectively. In LDA, the PV-CSDN scheme outperforms both DDCP and ANFIS by 96%. When PV-CSDN detects the attack, it sends a warning to the controller to shut down the malicious interface and stop the degradation of hit ratio. While in DDCP, the defense mechanism is based on identifying the malicious requests, adding them to an attack table, and sending a broadcast warning to neighbors to not cache the contents of these requests. This defense mechanism depends on the identification of the requests from the attacker, which means if any request is misidentified, the content of it will be cached and this will degrade the cache hit ratio of the router and decrease the network performance. ANFIS mitigates the attack by deleting the contents in a cache replacement algorithm that has a lower goodness value.

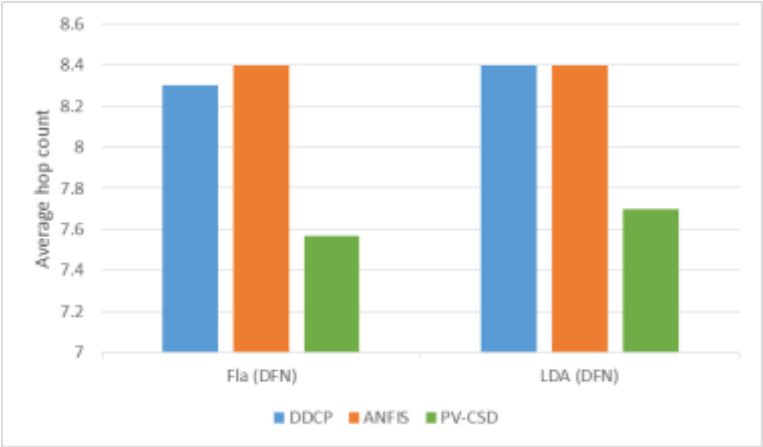



**Figure 14:** Average hop under CPA.

Figure 14 shows the average hop count for C3 in DFN topology during the attack. This consumer is the most affected by the attack because C3 is next to attacker A3. The PV-CSDN algorithm outperforms DDCP and ANFIS in FLA by 8.9% and 9.8%, respectively. In LDA, the PV-CSDN algorithm outperforms both DDCP and ANFIS by about 8.3%.

## 6. Conclusion

In this paper, PV-CSDN, a detection and defense scheme is proposed to prevent the effect of CPA in a CCN-SDN network. Two traffic characteristics have been used for the detection: the popularity of the contents and the average rate of repeated requests for each interface in the router. The algorithm is based on calculating the popularity of each content to use it in getting a threshold value that represents the popularity of the current traffic. In addition, calculating the variance of each content to find the average rate of repeated requests on each interface. The two threshold values will be used to find out if any abnormal behavior is being carried out in the network. If FLA or LDA is detected, a defense process is started by the controller to block the malicious interface and prevent any further damage to the network. The emulation results have demonstrated that the scheme performance has outperformed other schemes like DDCP and ANFIS, in cache hit ratio (up to 96% gain) and average hop count (up to 12% gain). Future work includes using this approach in larger and more complex networks. Besides, using machine language methods like support vector machines to classify the traffic and detect malicious requests.

## References

**[1]** Q. -Y. Zhang, X. -W. Wang, M. Huang, K. -Q. Li and S. K. Das, "Software Defined Networking Meets Information Centric Networking: A Survey," *IEEE Access*, vol. 6, pp. 39547-39563, 2018, doi: 10.1109/ACCESS.2018.2855135.

**[2]** Z. Zhang, C. -H. Lung, M. St-Hilaire and I. Lambadaris, "An SDN-Based Caching Decision Policy for Video Caching in Information-Centric Networking," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 1069-1083, April 2020, doi: 10.1109/TMM.2019.2935683.

**[3]** G. Siracusano, S. Salsano, P.L. Ventre, A. Detti, O. Rashed and N. Blefari-Melazzi, "A framework for experimenting ICN over SDN solutions using physical and virtual testbeds," *Computer Networks*, vol. 134, pp. 245 - 259, 2018, doi:10.1016/j.comnet.2018.01.026.

**[4]** Z. Xu, B. Chen, N. Wang, Y. Zhang and Z. Li, "ELDA: Towards efficient and lightweight detection of cache pollution attacks in NDN," *in 2015 IEEE 40th Conference on Local Computer Networks (LCN)*, pp. 82-90, 2015, doi: 10.1109/LCN.2015.7366286.

**[5]** H. Song, "Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane, " *in 2013 Proc. of SIGCOMM HotSDN Workshop (HotSDN'13)*, Hong Kong, China, Aug. 2013, doi:10.1145/2491185.2491190.

**[6]** P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese and D. Walker, "P4: Programming Protocol-independent Packet Processors," in 2014 ACM SIGCOMM Comput. Commun. Rev., vol. 44, pp. 87–95, July 2014, doi:10.1145/2656877.2656890.

**[7]** P. Ma, J. You, L. Wang and J. Wang, "Source routing over protocol oblivious forwarding for named data networking," *Journal of Network and Systems Management*, vol. 26, no. 4, pp. 857–877, 2018, doi:10.1007/s10922-017-9445-9.

**[8]** D. Hu, S. Li, H. Huang, W. Fang and Z. Zhu, "Flexible Flow Converging: A Systematic Case Study on Forwarding Plane Programmability of Protocol-Oblivious Forwarding (POF)," IEEE Access, vol. 4, pp. 4707-4719, 2016, doi: 10.1109/ACCESS.2016.2600619.

**[9]** S. Charpinel, C. A. Saibel Santos, A. B. Vieira, R. Villaca and M. Martinello, "SDCCN: A Novel Software Defined Content-Centric Networking Approach," in 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), 2016, pp. 87-94, 2016 ,doi: 10.1109/AINA.2016.86.

**[10]** Y. Gao, L. Deng, A. Kuzmanovic and Y. Chen, "Internet Cache Pollution Attacks and Countermeasures," *in Proceedings of the 2006 IEEE International Conference on Network Protocols*, pp. 54-64, 2006, doi: 10.1109/ICNP.2006.320198.

**[11]** L. Deng, Y. Gao, Y. Chen, and A. Kuzmanovic, "Pollution attacks and defenses for internet caching systems," *Computer Networks the International Journal of Computer &*

*Telecommunications Networking*, vol. 52, no. 5, pp. 935–956, April 2008, doi:10.1016/j.comnet.2007.11.019.

**[12]** A. Karami and M. Guerrero-Zapata, "An ANFIS-based cache replacement method for mitigating cache pollution attacks in Named Data Networking," *Computer Networks*, vol. 80, pp. 51–65, 2015, doi: 10.1016/j.comnet.2015.01.020.

**[13]** G. Zhang, J. Liu, X. Chnag and Z. Chen, "Combining Popularity and Locality to Enhance In-Network Caching Performance and Mitigate Pollution Attacks in Content-Centric Networking," *IEEE Access*, vol. 5, pp. 19012-19022, 2017, doi: 10.1109/ACCESS.2017.2754058.

**[14]** L. Yao, Y. Zeng, X. Wang, A. Chen and G. Wu, "Detection and Defense of Cache Pollution Based on Popularity Prediction in Named Data Networking," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2848-2860, Nov.-Dec. 2021, doi: 10.1109/TDSC.2020.2967724.

**[15]** J. Chen, L. Yue, J. Chen, "Mitigating cache pollution attacks in information centric mobile internet," *KSII Transactions on internet and information systems*, vol. 13 , no. 11, pp. 5673 - 5691, 2019, doi:10.3837/tiis.2019.11.022.

**[16]** P.V. Rani, S.M. Shalinie, "FuRL: fuzzy RBM learning framework to detect and mitigate network anomalies in Information Centric Network," *Sādhanā*, vol. 45, no. 1, pp. 5673 – 5691, May 2020, doi:10.1007/s12046-020-01331-3.

**[17]** L. Yao, Z. Fan, J. Deng, X. Fan and G. Wu, "Detection and Defense of Cache Pollution Attacks Using Clustering in Named Data Networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 6, pp. 1310-1321, Nov.-Dec. 2020, doi: 10.1109/TDSC.2018.2876257.

**[18]** T. Kamimoto, K. Mori, S. Umeda, and Y. Ohata, "Cache protection method based on prefix hierarchy for content-oriented network," *IEEE Consumer Communications & NETWORKING Conference*, pp. 417–422, 2016.

**[19]** M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," *Proc. of the International Conference on Computer Communications (INFOCOM'12)*, vol. 131, no. 5, pp. 2426–2434, 2012.

**[20]** M. Conti, P. Gasti, and M. Teoli, "A lightweight mechanism for detection of cache pollution attacks in named data networking," *Computer Networks*, vol. 57, no. 16, pp. 3178–3191, 2013, doi: 10.1016/j.comnet.2013.07.034.

**[21]** L. Breslau, Pei Cao, Li Fan, G. Phillips and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," *in 1999 IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, pp. 126-134 vol.1, 1999, doi: 10.1109/INFCOM.1999. 749260.

**[22]** B. Lantz and B. Heller, "Mininet: rapid prototyping for Software Defined Networks," 2021. [Online]. Available: http://mininet.org/.

**[23]** Content-Centric Networking CCNx Reference Implementation, "Official implementation of the ccn model," 2021. [Online]. Available: https://github.com/ProjectCCNx/ccnx.