Analysis, Detection and Mitigation of Cache Pollution Attack and Interest Flooding Attack in Named Data Networking

Chinmay Pani, Himanshu Bansla, Debasmita Das, Daniel Wintermeyer, Laxmikant Yadav, Abhishek

under the Guidance of Dr. Shashank Srivastava
Computer Science and Engineering Department

MNNIT ALLAHABAD

Outline I

Outline II

Motivation

The internet is changing

- How we use it?
- Why we use it?
- Increase in Multicast use
- Privacy and Security
- Limitations of the current architecture

What is NDN?

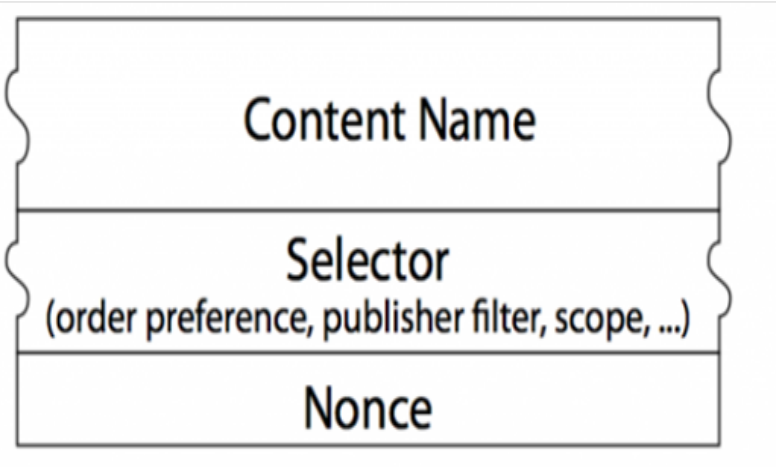
NDN is a very fast developing architecture that is designed to remove the limitations of the currently working IP based architecture. NDN retains the same hourglass architecture which allows upper and lower layers to function and develop independently. NDN identifies the data by naming them instead of identifying the end-systems, thus emphasizing on the quality of content of data instead of its source. It classifies the data into hierarchies based on their name prefixes.

Why NDN?

- Dominant paradigm shift towards **ICN (Information Centric Network)**.
- ICN represents a broad research direction of content/information/data-centric approach to network architecture. NDN is a specific architecture design under the broad ICN umbrella
- NDN is a clean-slate design in that it is a completely new architecture and has no dependency on IP.

Interest Packet

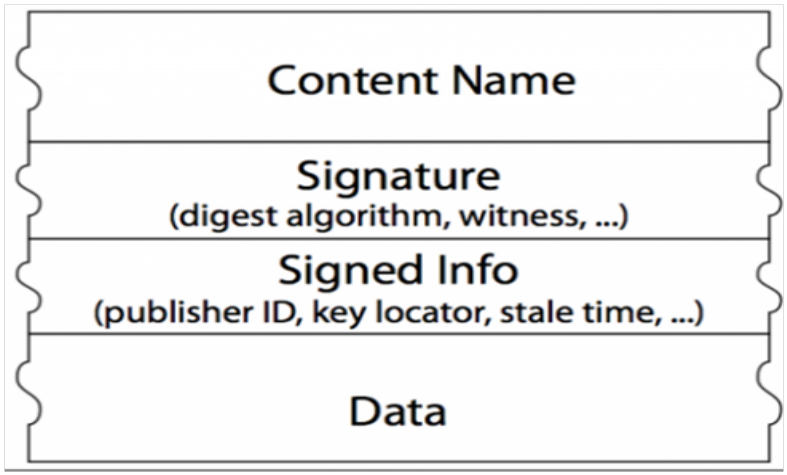
The Consumer sends out an interest packet which contains the name of the data required. Once the interest packet has been created, it follows a specific path through the Content Store, the Pending Interest Table and the Forwarding Information Base of the routers.



[2]

- **Content Name** is the object identifier.
- **Selector** is an optional filled indicating packet order preference etc.
- **Nonce** is a random number assigned by the PIT.

Data Packet



[2]

- **Content Name** is the object identifier.
- **Signature** is for security purpose.
- **Data** is the required data.

Routing Path

The Three Sections of the Routers are :-

- **Content Store (CS):**The router has some fix memory space to save those Data Packets which are recently used. When this space is fully occupied, then router replaces some Data packet which is not used for a long time. To replace those Data Packet router uses LRU (Least Recently Used), LFU (Least Frequently Used), etc. algorithms.
- **The Pending Interest Table (PIT):** PIT holds the entry records of the names of all pending interests and the interfaces from which the interests have been received.This allows the router to avoid forwarding the same interest packet individually for each consumer.

- **The Forwarding Information Base (FIB):** Whenever consumer sends an interest packet to the router, the router uses the FIB. FIB decides the path of interest packet to the server by using hierarchical naming for data packets. FIB stores two things about any interest packet:
  Prefix: The prefix contains the name of the data which is requested by the consumer.
  Interface: The interface contains the interface of the router at which the request arrives.
  On the basis of its prefix entry, FIB forwards the interest packet towards the data source.

Cache Pollution Attack

- NDN is primarily oriented towards large-scale content distribution. Due to this, every router provides an arbitrary amount of cache that store forwarded content on subsequent retrieval.
- However, this creates problems related to the caches like cache pollution attack.
- In a cache pollution attack, the main aim of the attacker is to force the NDN routers to cache the non-popular content.
- These attacks do not prevent the users from retrieving the content.

Types of Cache Pollution Attack

It is of 2 types:-

- **Disrupting Cache Locality:-** The attacker constantly issuing interest packets for non-popular contents of data, thus ruining the cache locality. Due to this, the adequacy of the cache is degraded, as popular content gets replaced by the non-popular content.
- **Creating a False Locality:-** The attacker continuously and repeatedly requests the same but small set of non-popular content of data.

Interest Flooding Attack

- Interest Flooding Attack or IFA is largely concerned with issuing a large number of Interests for non-existent content thereby flooding the Pending Interest Table (PIT) which can severely disrupt the network services by decreasing the throughput delivered to legitimate consumers.
- Interests whose aim is to fetch Data for a legitimate purpose are commonly referred to as Legitimate Interests (LIs). While Interests aiming to achieve network or producers service degradation are commonly referred to as Malicious Interests (MIs).
- Malicious Interests can be satisfied because they sometimes refer to the existing content or to dynamically generated content or can be fake if they refer to the non-existing content.

Advanced Interest Flooding Attacks [5]

- **blended IFA (bIFA):** In bIFA, attacker generate interest for both existent content and non-existent content. The main aim of a bIFA is to influence the detection metrics observed by routers, in order to both stay undetected and lower the probability for malicious Interests to be dropped. This causes countermeasures relying on satisfaction ratio as their detection mechanism to fail.
- **chameleonic IFA (cIFA):** cIFA includes the attackers which change the target prefix name after a certain time window. It tries to avoid the various countermeasures taken by the routers where few of the prefix names are marked as infected. This causes countermeasures relying on blacklisting certain prefixes used by attackers to fail.

Related Work

Though the field is relatively new, due to the promising future of NDN, a lot of work has already been done related to it.
Several countermeasures have been proposed against the traditional IFA.
We discuss one such popular countermeasure, which we have also implemented and analysed its performance against bIFA and cIFA.

Poseidon:- Poseidon[1] is used for detecting and mitigating interest flooding attacks. It consists of a detection and a reaction phase. Detection is either local or distributed. Poseidon is a set of algorithms that execute on routers to identify traffic anomalies (especially, interest flooding) and mitigate their effects. It continuously monitors per-interface rates of unsatisfied interests with respect to overall traffic. If the rates change significantly between two consecutive time intervals, it sets a filter on the offending interface(s) that reduces the number of incoming interests.

Related Work

Detection Parameters:- Attacks are detected using two parameters: $\omega(r_i^j, t_k)$ and $\rho(r_i^j, t_k)$. $\omega(r_i^j, t_k)$ is the number of incoming interests divided by the number of outgoing content packets, observed by a router r_i on its interface r_i^j within time interval t_k :

$$\omega(r_i^j, t_k) = \frac{number of interests from r_i^j at interval t_k}{number of content packets from r_i^j at interval t_k} \quad (1)$$

$\rho(r_i^j, t_k)$ is the number of bytes used to store interests in PIT, coming from interface r_i^j within time interval t_k.

Cache Pollution Attack

- **Goal** - Detection of the Cache Pollution Attack and mitigation or reducing its negative impact on the network.
- **Assumption** - The fraction of attackers in a network is less as compared to the consumers. This assumption is quite general and is the scenario in nearly all realistic attack scenarios.

- Our idea is that the number of distinct users requesting a name prefix becomes a very important factor in deciding the popularity of the name prefix.
- So, we collected global data about the number of distinct users for each prefix periodically over time and use this data to calculate the popularity rating of each prefix at that instant of time.
- The newly calculated rating is passed through a method called Exponentially Weighted Moving Average (EWMA)[3] which is used to calculate the subsequent popularity for each name prefix.

Algorithm 1 OnInterest (Trigger on interest packet retrieval)

INPUT: interest, inFace

```
1: hopCount = interest.hopCountTag
2: if hopCount = 1 then
3:     name = interest.name
4:     DRS.store(name, inFace)        ▷ DRS : Distinct Request Set
5: Usual interest process
```

- Let the total numbers of users (legitimate users and attackers together) in the network be n, the number of distinct users requesting a name prefix in the current time period Ti be xi, then the new rating Ri of the prefix for this interval is calculated as: $R_i = x_i/n$
- The initial EWMA[3] of each prefix A0 is set to 1.0. The later values for each time interval Ti (i = 1,..,n) is calculated as:

$$A_i = \alpha.R_i + (1 - \alpha).A_{i-1} \tag{2}$$

where α is the smoothing factor which lies in the open interval (0,1).

Algorithm 2 PeriodicPopularityUpdation (Trigger with a time period T_i)

```
1:  for each d ∈ DRS do
2:      name = d.name
3:      increment NDR[name] by 1          ▷ NDR : Number of Distinct Requests
4:  clearDRS
5:  for each name ∈ NDR do
6:      currentRating = NDR[name]/userCount
7:      if name ∈ PRD then                ▷ PRD : Popularity Rating Data
8:          pastRating = PRD[name]
9:      else
10:         pastRating = 1.0
11:     PRD[name] = α * currentRating + (1 − α) * pastRating
```

- This updated EWMA value is then used for making caching decisions in the next time interval.
- We have decided upon a threshold value Th. This is selected by hit and trial for a network and usually lies between 0.1 to 0.4 for most networks.
- Now, whenever a router receives a data packet:
  1. If the cache is not full, the data is cached without taking its EWMA popularity rating into consideration.
  2. If the cache is full, the decision of whether to cache the data is taken on the basis of whether the EWMA popularity lies above or below the threshold Th value.

Algorithm 3 OnData (Trigger on data packet retrieval, implies a cache miss)

INPUT: data, inFace

```
1: name = data.name
2: if name ∉ PRD then
3:     PRD[name] = 1.0
4: if PRD[name] > Th then
5:     Add data to CS                    ▷ CS : Content Store
6: Usual data process
```

Interest Flooding Attack

- Our primary goal is simulate the advanced interest flooding attacks namely-
  1. The blended IFA (bIFA)
  2. The chameleonic IFA (cIFA)

  and analyse their effects on realistic networks.
- We also aim to analyse how the currently used countermeasures against interest flooding attacks fare against the advanced ones.

- For performing interest flooding attack, we needed a stream of distinct non-existing interest packets.
- So we made i^{th} producer listen to prefix $/producer_i$, but only reply to interest packets with prefix $/producer_i/data$.
- This ensured that the only the first part of the prefix i.e., $/producer_i$ is stored in the forwarding information base (FIB) of routers which results in any interest starting with the first part of the prefix to be routed all the way to the respective producers.
- This results in new entries in PITs of all routers along the way, thus maximizing the impact of the attack.

- To make the non-existing interest packets unique, so that new PIT entries are created each request, we have used a non existing prefix of the format $/producer_i/notdata/j_i$, where j_i is a ever increasing counter per producer prefix.

Algorithm 4 SendPacket (Triggers based on attacker request frequency)

```
prefix = "/producer"
prefixNo = getRandomInt(1, np)                ▷ np: number of producers
prefix.append(prefixNo)
decisionRatio = getRandomDouble(0.0, 1.0)
if decisionRatio < existingPrefixRequestRatio then   ▷ Set to 0.0 for cIFA
    prefix.append("/data")
else
    prefix.append("/notdata")
counter[prefixNo] = counter[prefixNo] + 1
seqNo = counter[prefixNo]
prefix.append(seqNo)
Usual packet creation process
Usual packet sending process
```

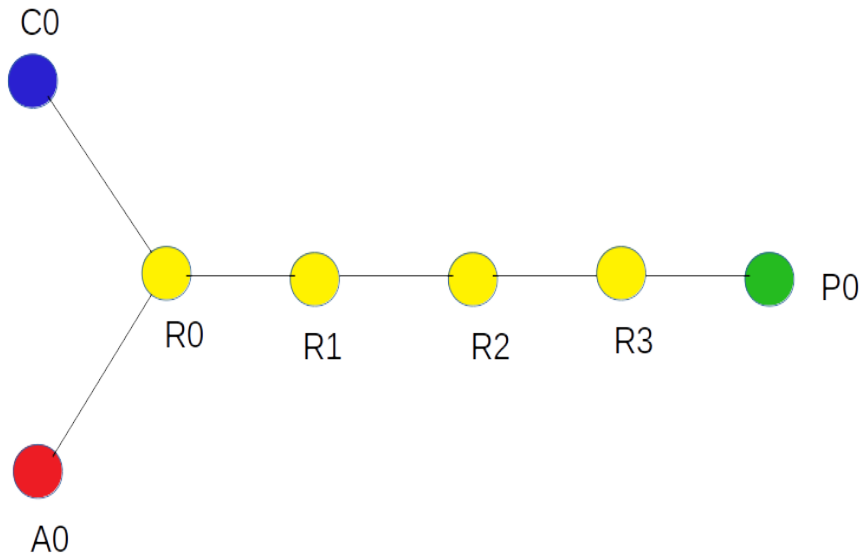
- Most of the currently used countermeasures rely on satisfaction ratio or prefix based blacklisting or a combination of both. While these work well against the traditional IFA, but they will fail against bIFA or cIFA.
- To demonstrate this, we have implemented one such popular countermeasure, Poseidon [1]. As stated earlier Poseidon relies on two parameters to detect an ongoing attack-
  1. Satisfaction Ratio per interface
  2. PIT usage per interface
- Now since in bIFA the attacker requests a mix of both existing and non-existing interests, this would up the satisfaction ratio and hence, Poseidon should fail to counter this attack in realistic networks. As far as cIFA is concerned, Poseidon may be able to detect and mitigate it in some cases.

Experimental Setup and Results Analysis

We have used **ns-3** simulator for testing out the idea that the existing countermeasures against Interest Flooding Attack are not effective against the newly discovered variations namely, bIFA and cIFA. First, we implemented the attacks and then have used the method described in Poseidon to counter it.
We have considered three popular topologies for our simulation-

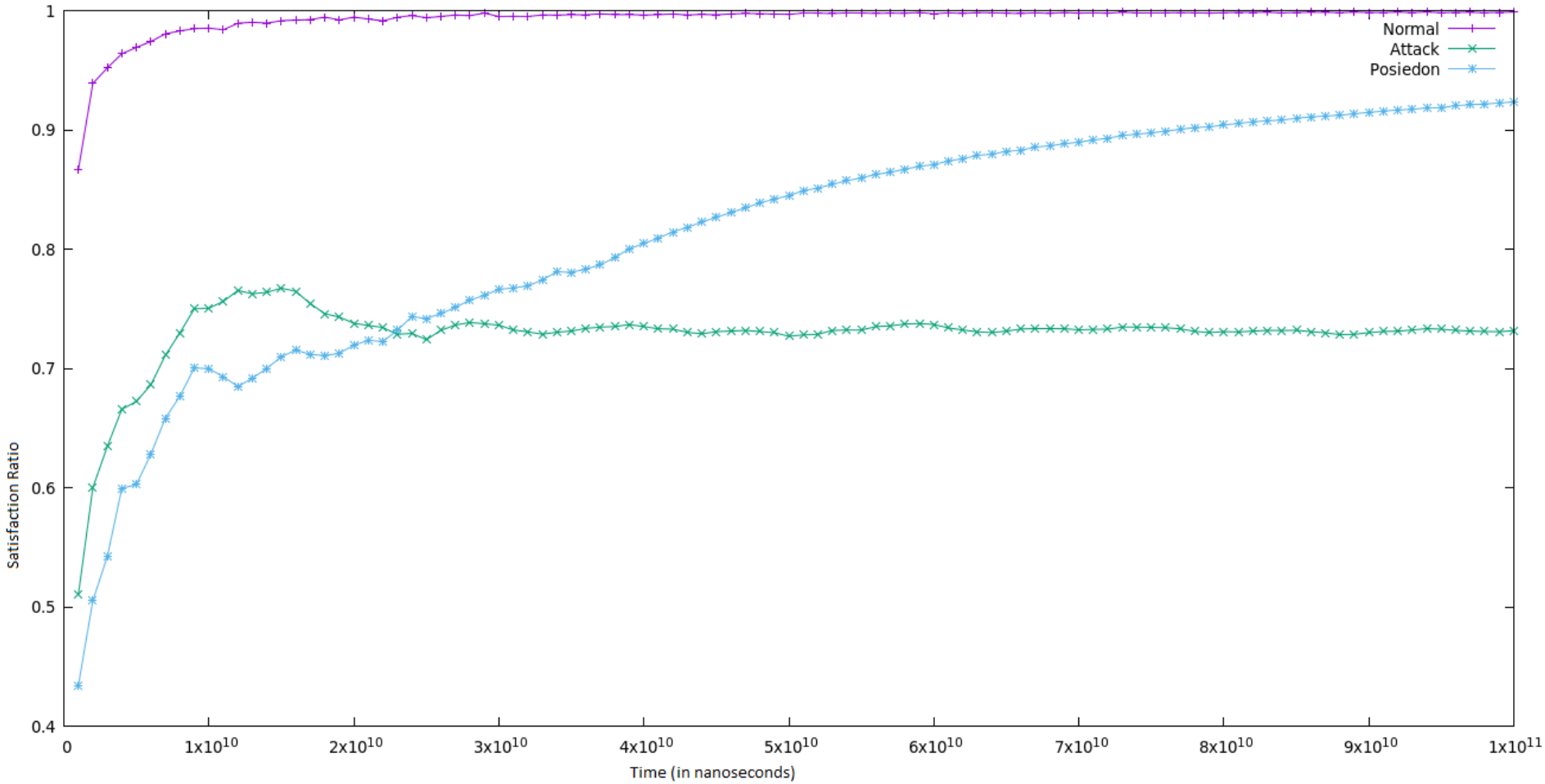
1. Linear Topology
2. Xie Complex (XC) topology[2]
3. DFN topology[2]

Linear Topology

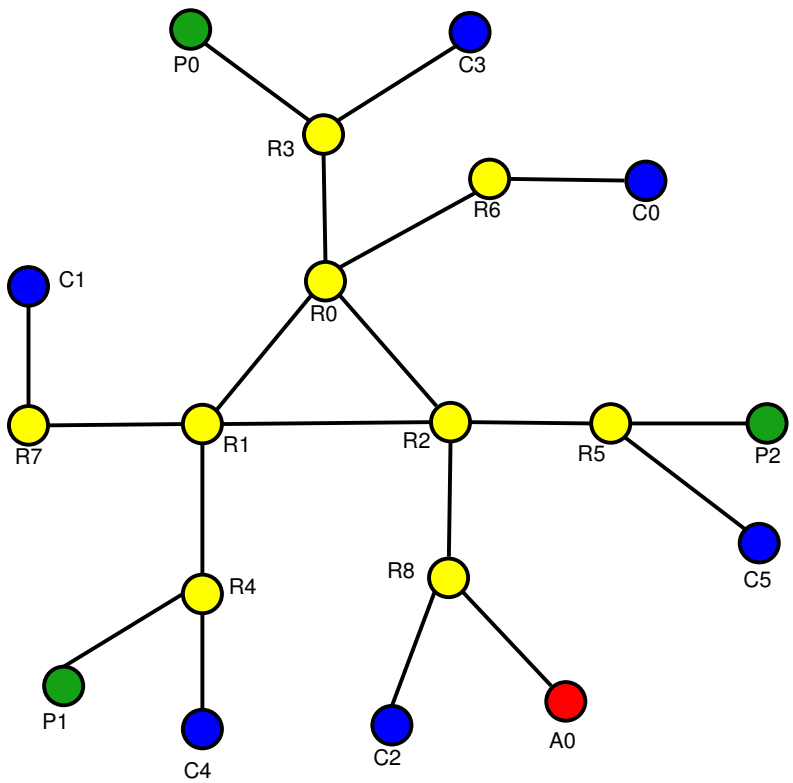


- Consists of one attacker, one consumer and one producer.
- Used simple topology to analyse the efficacy of the IFA attacks and the used countermeasure.
- Poseidon is able to improve performance against the attack.
- Due to the simple structure of the topology the attacker interface is easily identified.

IFA and Poseidon in Linear Topology



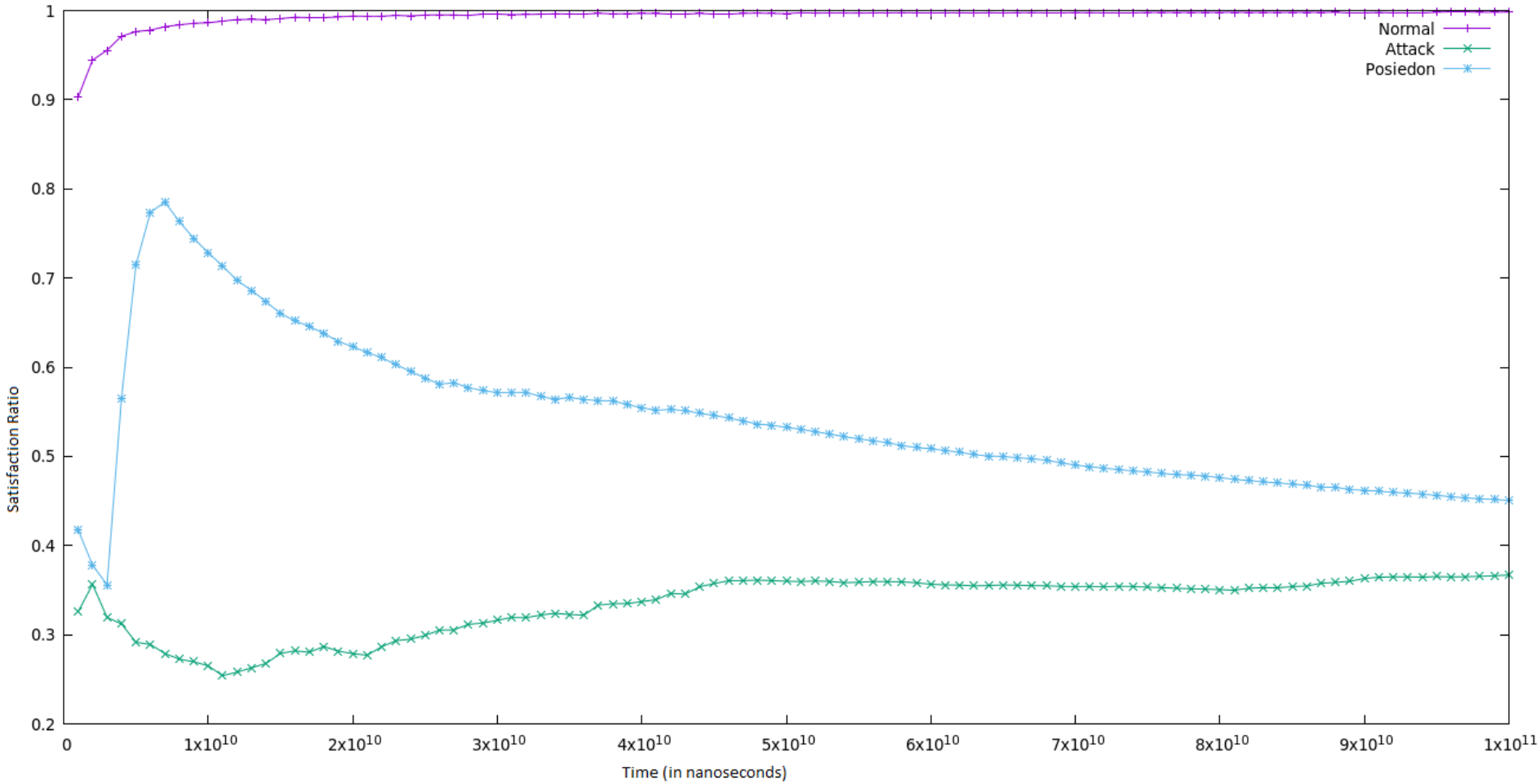
Xie Complex Topology



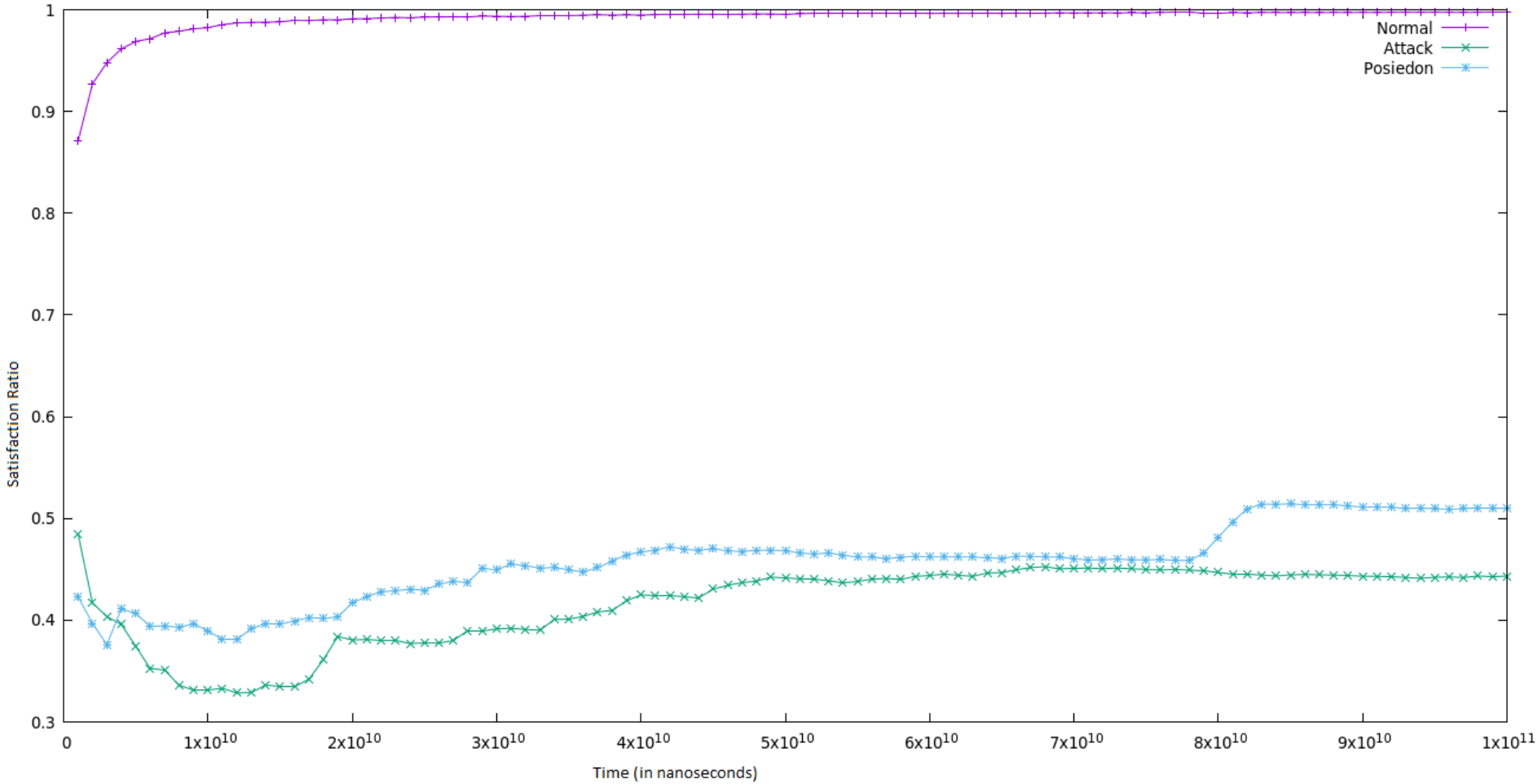
[2]

- Consists of one attacker and several consumers.
- Attackers frequency is set to 20 times the average consumers frequency.
- $satisfactionThreshold = 10.0$ and $pitSpaceThreshold = 0.15$.
- Poseidon is able to reduce the impact of the impact of the attack to some extent in cIFA, but fails in the case of bIFA as we claimed.

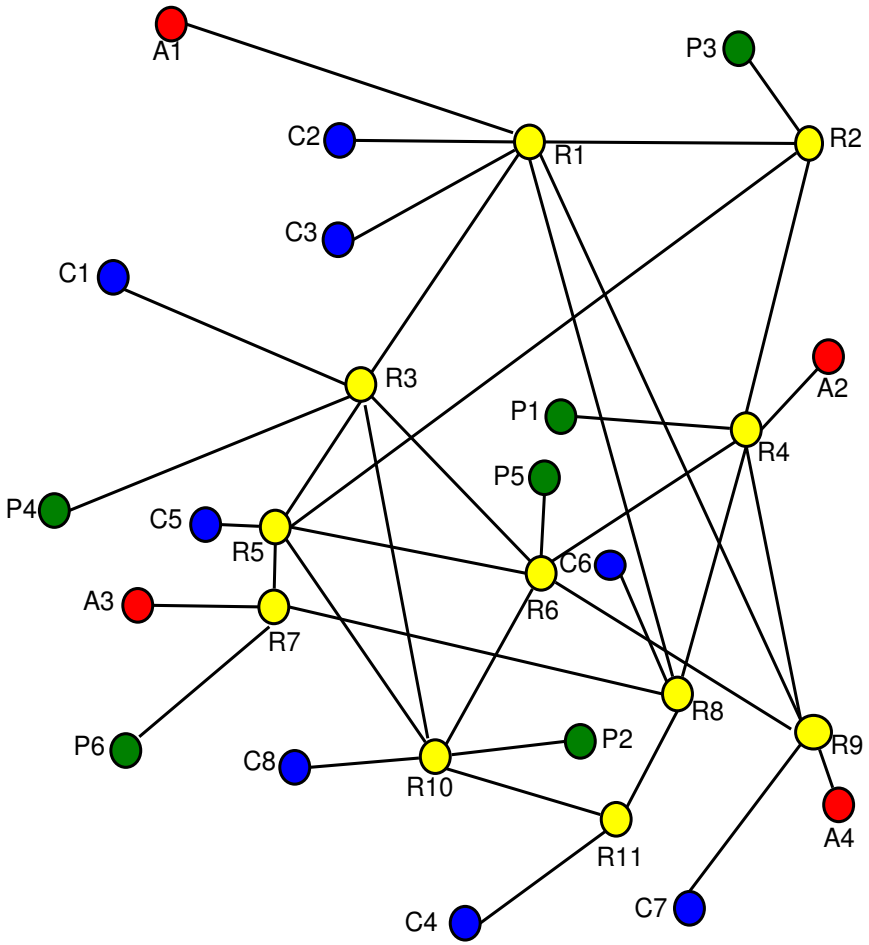
Satisfaction Ratios in XC Topology (cIFA)



Satisfaction Ratios in XC Topology (bIFA)



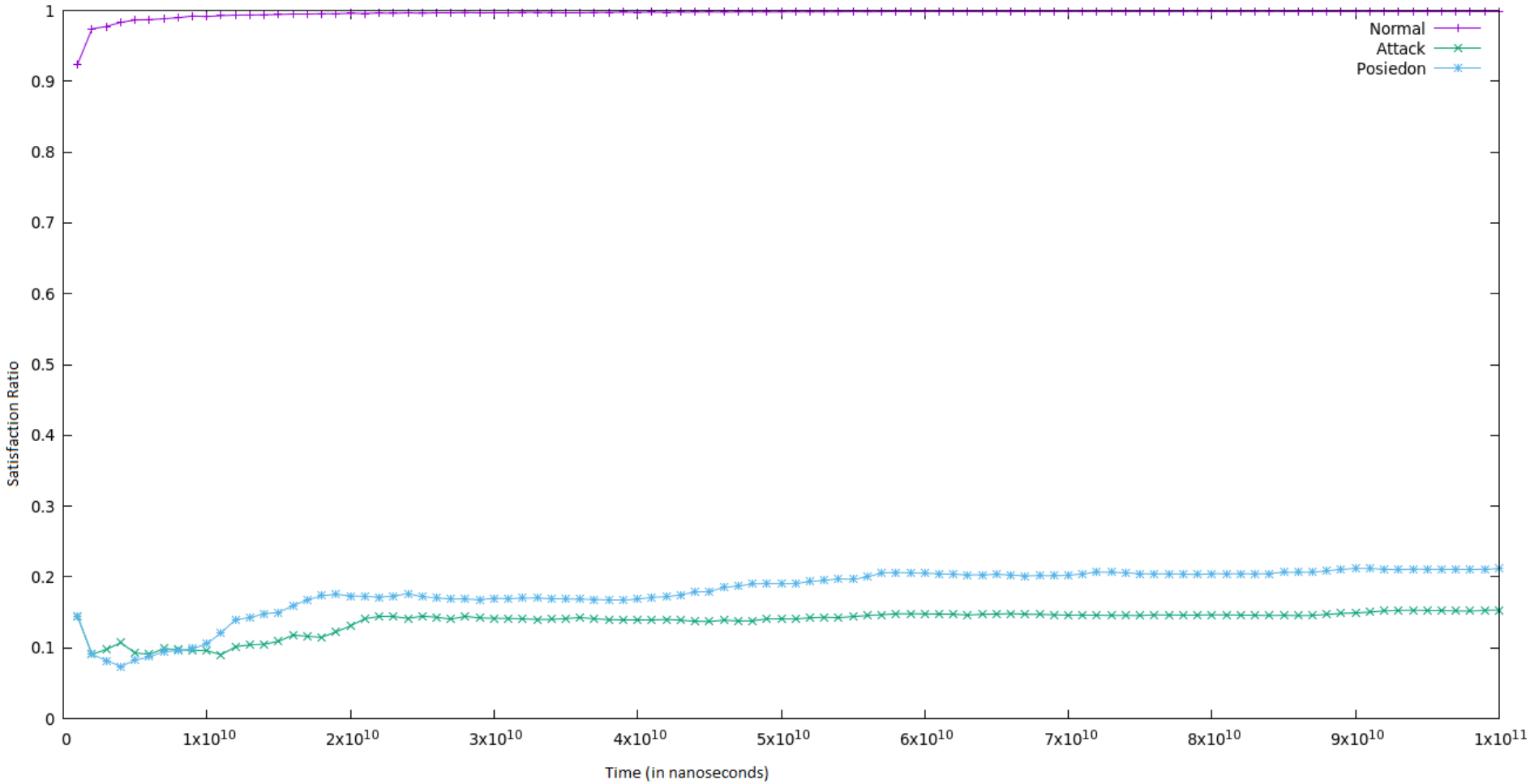
DFN Topology



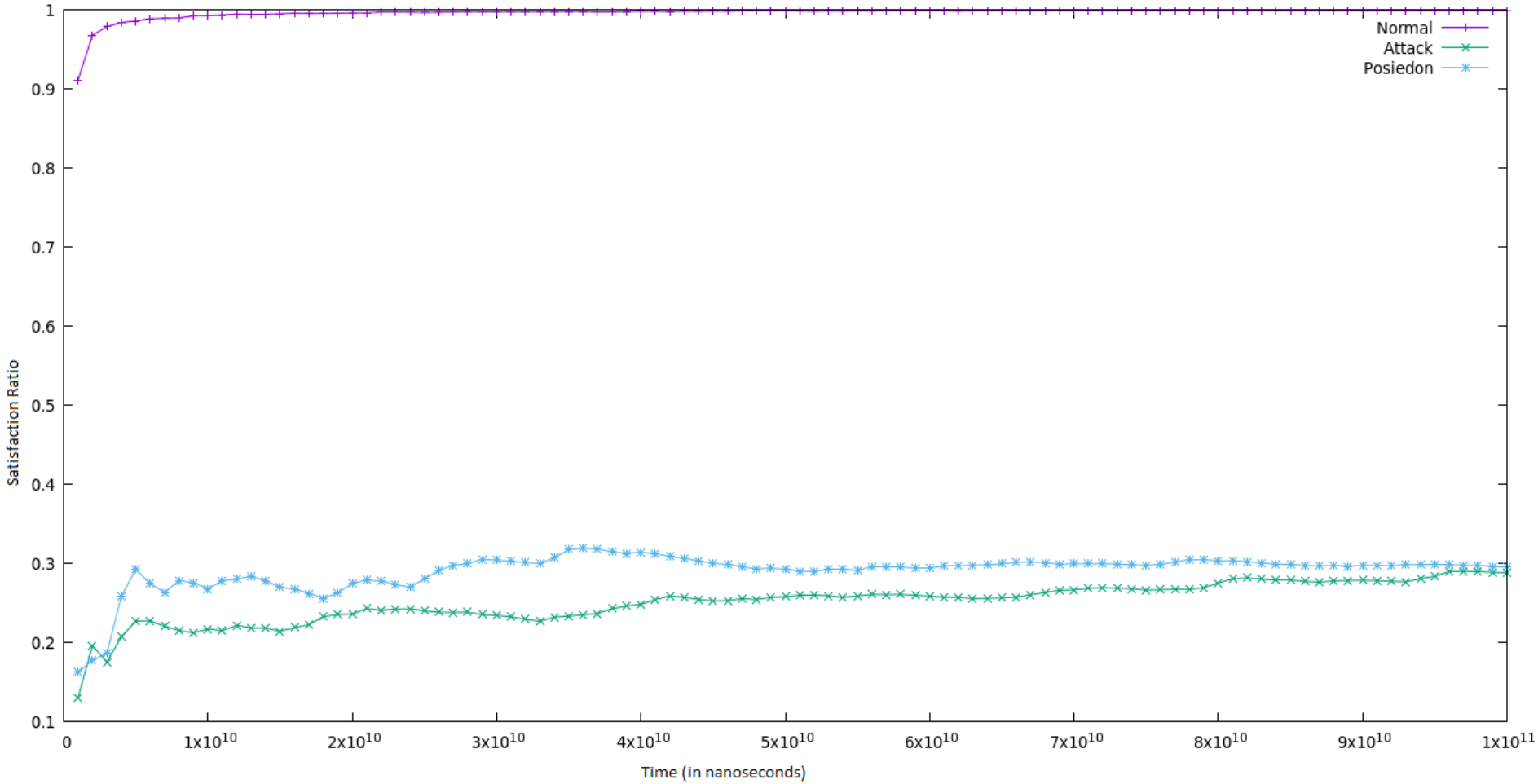
[2]

- Consists of several consumers and as many as 4 attackers.
- Attackers frequency is set to 5 times that of the average consumers.
- $satisfactionThreshold = 20.0$ and $pitSpaceThreshold = 0.10$.
- In this scenario Poseidon fails to provide any significant advantage in both cases.

Satisfaction Ratios in DFN Topology (cIFA)



Satisfaction Ratios in DFN Topology (bIFA)



Result Analysis

Since in bIFA the attacker requests a mix of both existing and non-existing interests, this would up the satisfaction ratio and hence, Poseidon fails to counter this attack in realistic networks. As far as cIFA is concerned, Poseidon is able to detect and mitigate to a small extent.

Hence, the relative comparison in the graphs shows that while Poseidon and other similar currently used countermeasures do well against traditional IFA, but fail to improve performance significantly in case of bIFA and cIFA.

Comparison with Other Cache Protection Methods

We have compared our work with two other cache protection methods:-

- Cache Protection Method Based on Prefix Hierarchy by Kamimoto et al[4] .
- Enhancing Cache Robustness: CacheShield by Xie et al [6].

Cache Protection Method Based on Prefix Hierarchy

It detects malicious user requests by observing the variation in request rate of the prefix being requested. Request rate p(i) of content i is calculated as:-

$$p(i) = \frac{n_r(i)}{\sum_{j \in S} n_r(j)}$$

It then blacklists the prefixes requested by attackers if the variation in request rate goes any beyond the Blacklist threshold(τ). This value may have to be adjusted according to the topology of the network. After experimenting with various values, we chose the $\tau = 0.1$, as it gave the best results for the topology we have used.

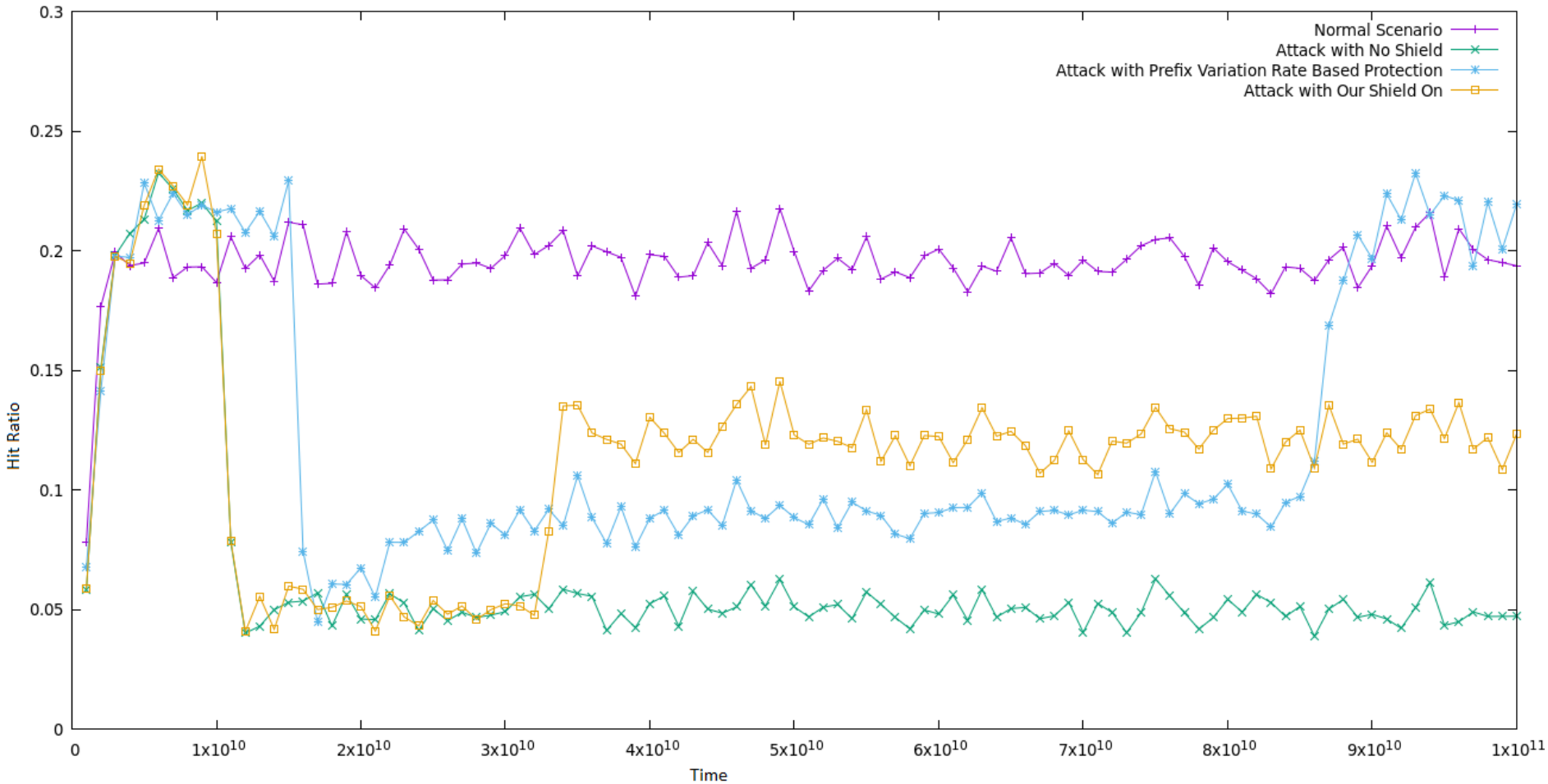


Figure: Comparison of Hit Ratios with Prefix Variation Rate Based Protection Method in DFN Topology

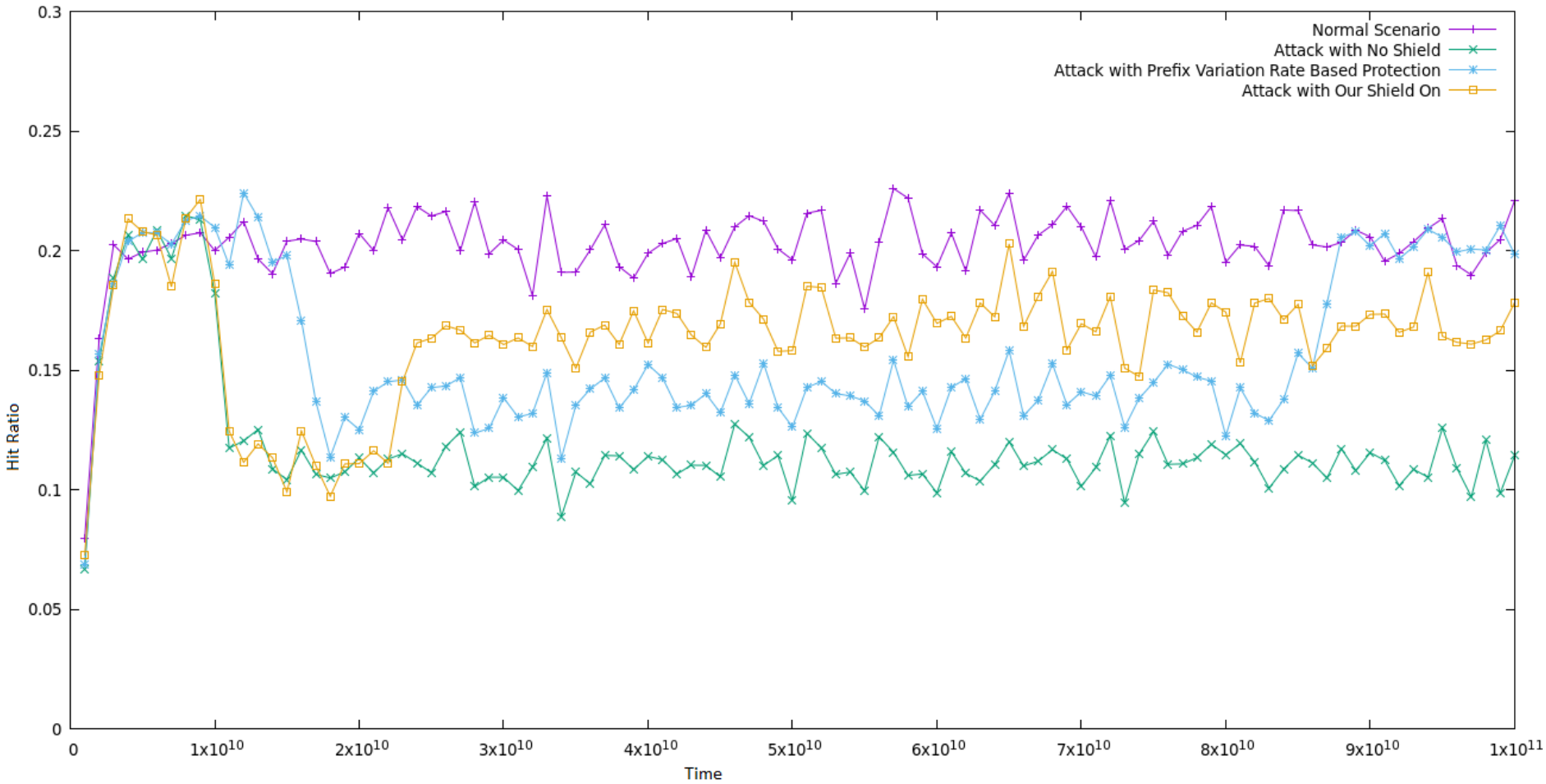


Figure: Comparison of Hit Ratios with Prefix Variation Rate Based Protection Method in XC Topology

Cache Shield Based Approach

This method works best against locality disruption attacks. Upon content hit, the requested item is sent back immediately. However, when a miss occurs, the decision to cache the data is made by using a shielding function.

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

Here, t denotes the t^{th} continuous miss for the specific content, whereas p and q are constants to be chosen according to the network. Upon experimentation we found the values $p = q = 50$ give the best results for our chosen topologies.

Figure: Comparison of Hit Ratios with CacheShield in DFN Topology

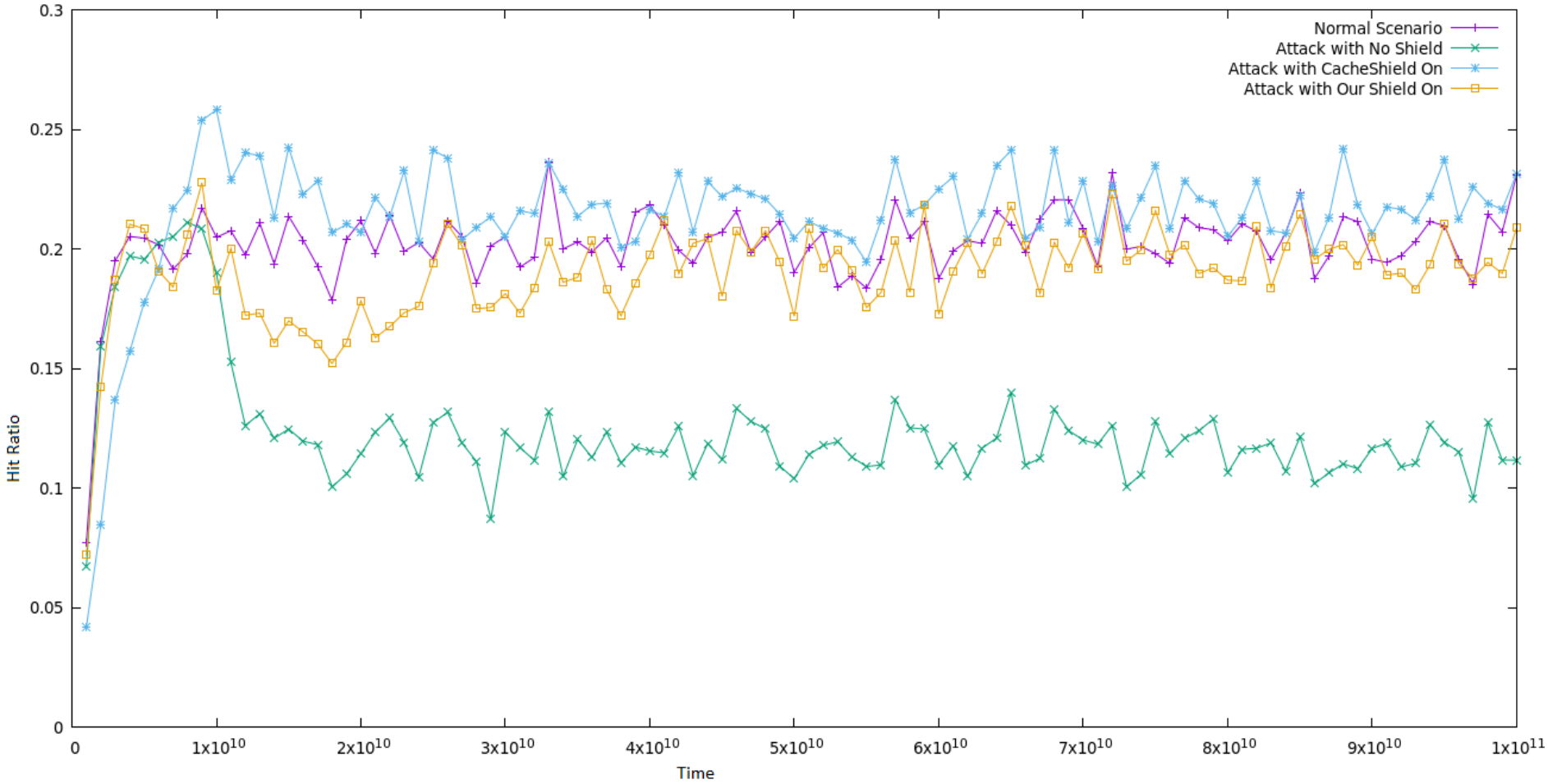


Figure: Comparison of Hit Ratios with CacheShield in XC Topology

References I

Compagno, A., Conti, M., Gasti, P., and Tsudik, G.
Poseidon: Mitigating interest flooding ddos attacks in named data networking.
In *38th annual IEEE conference on local computer networks* (2013), IEEE, pp. 630–638.

Conti, M., Gasti, P., and Teoli, M.
A lightweight mechanism for detection of cache pollution attacks in named data networking.
Computer Networks 57, 16 (2013), 3178–3191.

Hunter, J. S.
The exponentially weighted moving average.
Journal of quality technology 18, 4 (1986), 203–210.

Kamimoto, T., Mori, K., Umeda, S., Ohata, Y., and Shigeno, H.
Cache protection method based on prefix hierarchy for content-oriented network.
In *Consumer Communications & Networking Conference (CCNC), 2016 13th IEEE Annual* (2016), IEEE, pp. 417–422.

References II

Signorello, S., Marchal, S., François, J., Festor, O., and State, R.
Advanced interest flooding attacks in named-data networking.
In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)* (2017), IEEE, pp. 1–10.

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